Smart Contract Security Audit V1

Dream Token Smart Contract

10/2/2023

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 Information

- **Platform**: Binance Smart Chain
- **Contract Address**: 0xce92d12f231ee76e8751b5547ebc844b741e44bb
- **Code Source:** https://bscscan.com/address/0xce92d12f231ee76e8751b5547ebc844b741e44bb#code

Contracts address deployed to test net (Binance)

Dream Token smart contracts on Binance test-net by the auditor to test every function .

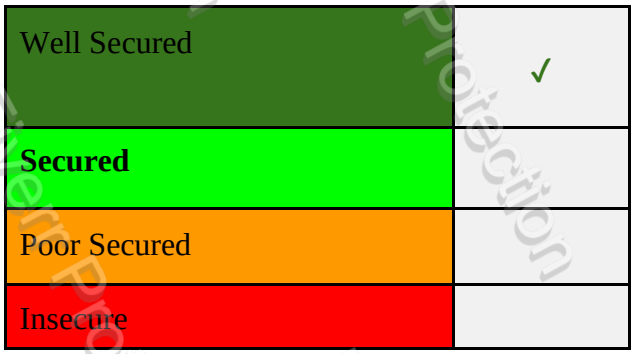
https://testnet.bscscan.com/address/0xbea7301c86934488ac0b45986a388b506812a0f4

Token Information:

Name	Dream
Symbol	WIN
Total supply	1,0000,000,000
Decimals	18
Marketing Fee	2%
Liquidity Fee	2%
Devolvement Fee	2%
Marketing Wallet	0xaE7844a5B9325d7B4Bf11EBa223f867E3b29cDD9
Devolvement Wallet	0xB9EEE71FfA3e50E39c377BD014b5212E93e15E22
Dead wallet	0x000000000000000000000000000000000000dEaD
Number of token to sell to add liquidity	5000000
PCS V2 Pair	0x38456886D70dC01F5c6b188a660BC869C2cfc169
PCS V2 Router	0x10ED43C718714eb63d5aA57B78B54704E256024E

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well Secured**. The team fix the high issue.



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

The files:

Dream.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Dream.sol	**2135dc59ffe59fd874a54b8deeaeba4c77585efb626df8865bf847109fb23457**	0xce92d12f231ee76e8751b5547ebc844b741e44bb

- Contract: Dream
- Inherit: Ownable, IBEP20
- 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**
owner	✓	Read / public	**Passed**
_ totalSupply	✓	Read / public	**Passed**
DeadWalletAddress	✓	Read / public	**Passed**
MarketingWalletAddress	✓	Read / public	**Passed**

DevelopmentWalletAddress	✓	Read / public	**Passed**
isExcludedFromTax	✓	Read / public	**Passed**
swapAndLiquifyEnabled	✓	Read / public	**Passed**
numTokensSellToAddToLiquidity	✓	Read / public	**Passed**
TotalTaxFee	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
excludeFromfee	✓	Write / public	**Passed**
includeFromfee	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
setSwapAndLiquifyEnabled	✓	Write / public	**Passed**
UpdateNoOfTokensSellToGetReward	✓	Write / public	**Passed**
UpdateTaxFees	✓	Write / public	**Passed**
UpdateWallets	✓	Write / public	**Passed**
transferOwnership	✓	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 with notes**
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:

#Logic errors

Description

This smart contract has devolvement and marketing fees it should go to 2 addresses but after the auditor tests it all fees go to the smart contract address, not to the wallets address, and you can check these transactions:

https://testnet.bscscan.com/tx/0x395815d1b03c2f700cfc6f8474bb79d9715bda0084971325c84d0d66d5d328cb

https://testnet.bscscan.com/tx/0x259971cd94ff51d7159f6378bf9972cdf8fec99821cda5df506e8c2f93a725f1

the code:

```
uint256 amountReceived = amount - (totalTax);
        _balances[address(this)] += totalTax;
        _balances[sender] = _balances[sender] - amount;
        _balances[recipient] += amountReceived;
        if (totalTax > 0) {
            emit Transfer(sender, address(this), totalTax);
        }
        emit Transfer(sender, recipient, amountReceived);
    }
```

Remediation

Redesign the function and make it send the fees to devolvement wallet and marketing wallet.

Status: Closed. Fixed in version 2, and you can check here

https://testnet.bscscan.com/tx/0x4625f2b4f9f047c30517c906d81f5b2ee861eb5b0c5d01661a181ad4010c64ad

Medium:

No Medium severity vulnerabilities were found.

Low:

#Use of block.timestamp for comparisons

Description

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

Remediation
Avoid use of block.timestamp

Status: Acknowledged

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Naming Conventions

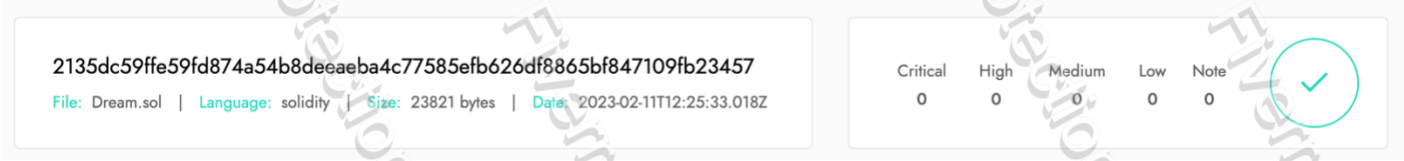
Description
The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names "__totalSupply " which is public.

Remediation
Remove "_" from external variable names and add it to private variable names.

Status: Acknowledged

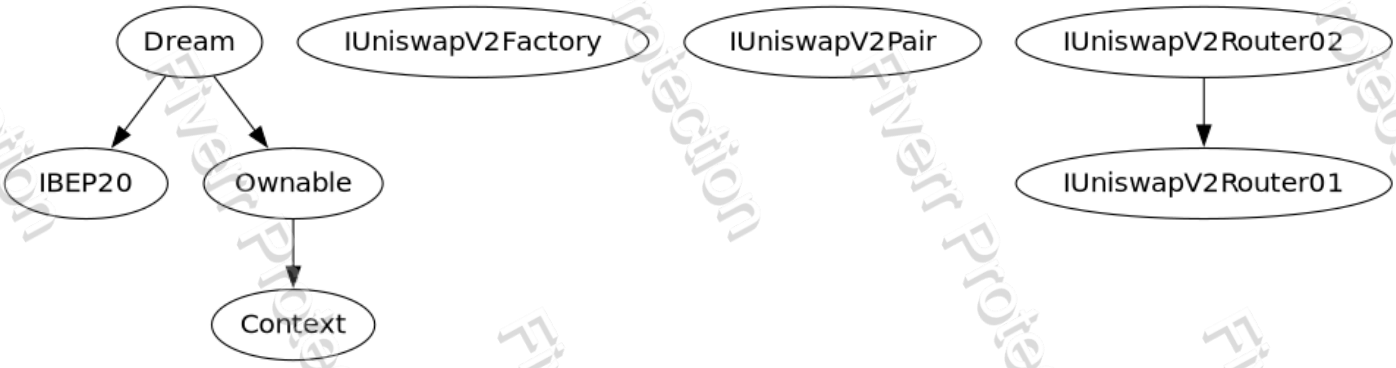
Automatic Testing

1- Check for security



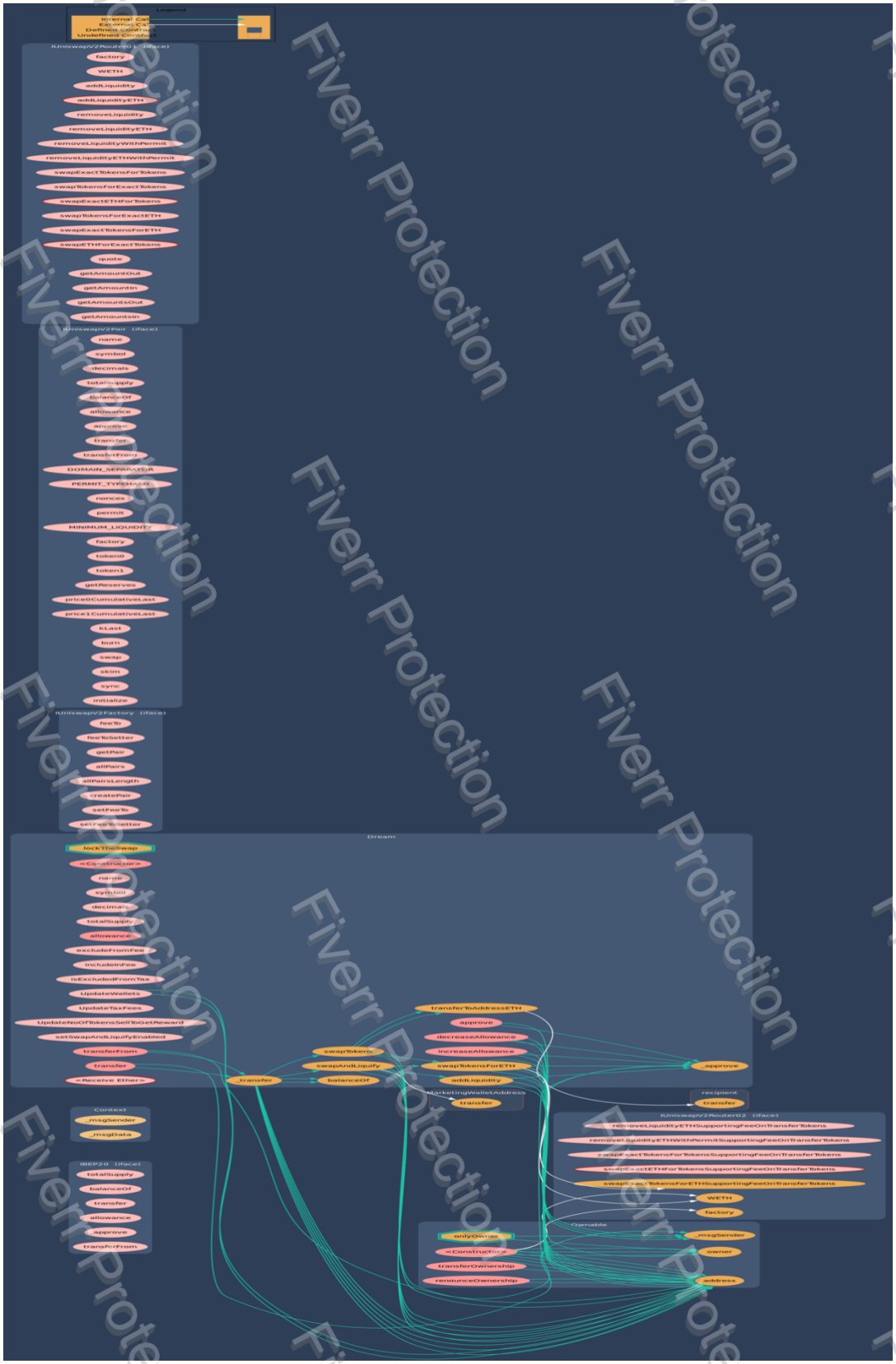
2- SOLIDITY STATIC ANALYSIS

3- Inheritance graph



4- SOLIDITY UNIT TESTING

5- Call graph



Unified Modeling Language (UML)

Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
ad5c4648  =>  WETH()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,b
ool,uint8,bytes32,bytes32)
ded9382a  =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,u
int8,bytes32,bytes32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
```

```
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
85f8c259  =>  getAmountIn(uint256,uint256,uint256)
d06ca61f  =>  getAmountsOut(uint256,address[])
1f00ca74  =>  getAmountsIn(uint256,address[])
af2979eb  =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,add
ress,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,u
int256,address,uint256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
b6f9de95  =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint25
6)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],addres
s,uint256)
104e81ff  =>  _approve(address,address,uint256)
a457c2d7  =>  decreaseAllowance(address,uint256)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
cb4ca631  =>  isExcludedFromTax(address)
b21b3fdc  =>  UpdateWallets(address,address)
92e6a163  =>  UpdateTaxFees(uint256,uint256,uint256)
9429b9fe  =>  UpdateNoOfTokensSellToGetReward(uint256)
c49b9a80  =>  setSwapAndLiquifyEnabled(bool)
30e0789e  =>  _transfer(address,address,uint256)
fe784eaa  =>  swapTokens(uint256)
4a8da1c5  =>  transferToAddressETH(address,uint256)
173865ad  =>  swapAndLiquify(uint256)
e56a645e  =>  swapTokensForETH(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Dream.sol	12b52c5974967be6d7d33e2a6b2cd3ea3aab6b01

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🔘	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🔘	NO❗
└	transferFrom	External ❗	🔘	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🔘	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🔘	onlyOwner
└	transferOwnership	Public ❗	🔘	onlyOwner
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🔘	NO❗
└	setFeeTo	External ❗	🔘	NO❗
└	setFeeToSetter	External ❗	🔘	NO❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🔘	NO❗
└	transfer	External ❗	🔘	NO❗
└	transferFrom	External ❗	🔘	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗

└	decreaseAllowance	Public ❗	🛑	NO❗
└	excludeFromFee	External ❗	🛑	onlyOwner
└	includeInFee	External ❗	🛑	onlyOwner
└	isExcludedFromTax	External ❗		NO❗
└	UpdateWallets	External ❗	🛑	onlyOwner
└	UpdateTaxFees	External ❗	🛑	onlyOwner
└	UpdateNoOfTokensSellToGetReward	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	External ❗	🛑	onlyOwner
└	transfer	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	<Receive Ether>	External ❗	💵	NO❗
└	_transfer	Internal 🔒	🛑	
└	swapTokens	Private 🔐	🛑	lockTheSwap
└	transferToAddressETH	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForETH	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	

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”. After fixing the logic errors.

✓ No mint function.
✓ 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.