



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Cairo Finance Audit

Security Assessment
12. May, 2022

For



CAIRO
FINANCE

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	20
Source Units in Scope	23
Critical issues	24
High issues	24
Medium issues	24
Low issues	24
Informational issues	24
Commented Code exist	28
Audit Comments	28
SWC Attacks	29

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Uniswap, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	10. May 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://cairo.finance/>

Telegram

<https://t.me/cairofinance>

Twitter

<https://twitter.com/cairofinance?s=11>

Instagram

<https://instagram.com/cairo.finance.official>

Youtube

<https://www.youtube.com/channel/UCKqHSmte97IizdLIRtdabYw>

Description

Cairo is a Yield Optimizer with his own deflationary staking system that focuses on safety and autocompounds crypto assets for the best APYs through the Binance Smart Chain.

Project Engagement

During the 6th of May 2022, **Cairo Finance Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link v1.0

- Github
 - Token
 - <https://github.com/cairofinance/cairo-contracts/blob/master/contracts/token/CairoToken.sol>
 - Commit: f8be3b5c866899a2604180673fbecb5dc422e4f9
 - Maximizer
 - <https://github.com/cairofinance/cairo-contracts/blob/master/contracts/maximizer/CairoMaximizer.sol>
 - Commit: ce4b5376654021e537ee3ef45bacd4dfaeda3bbc

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

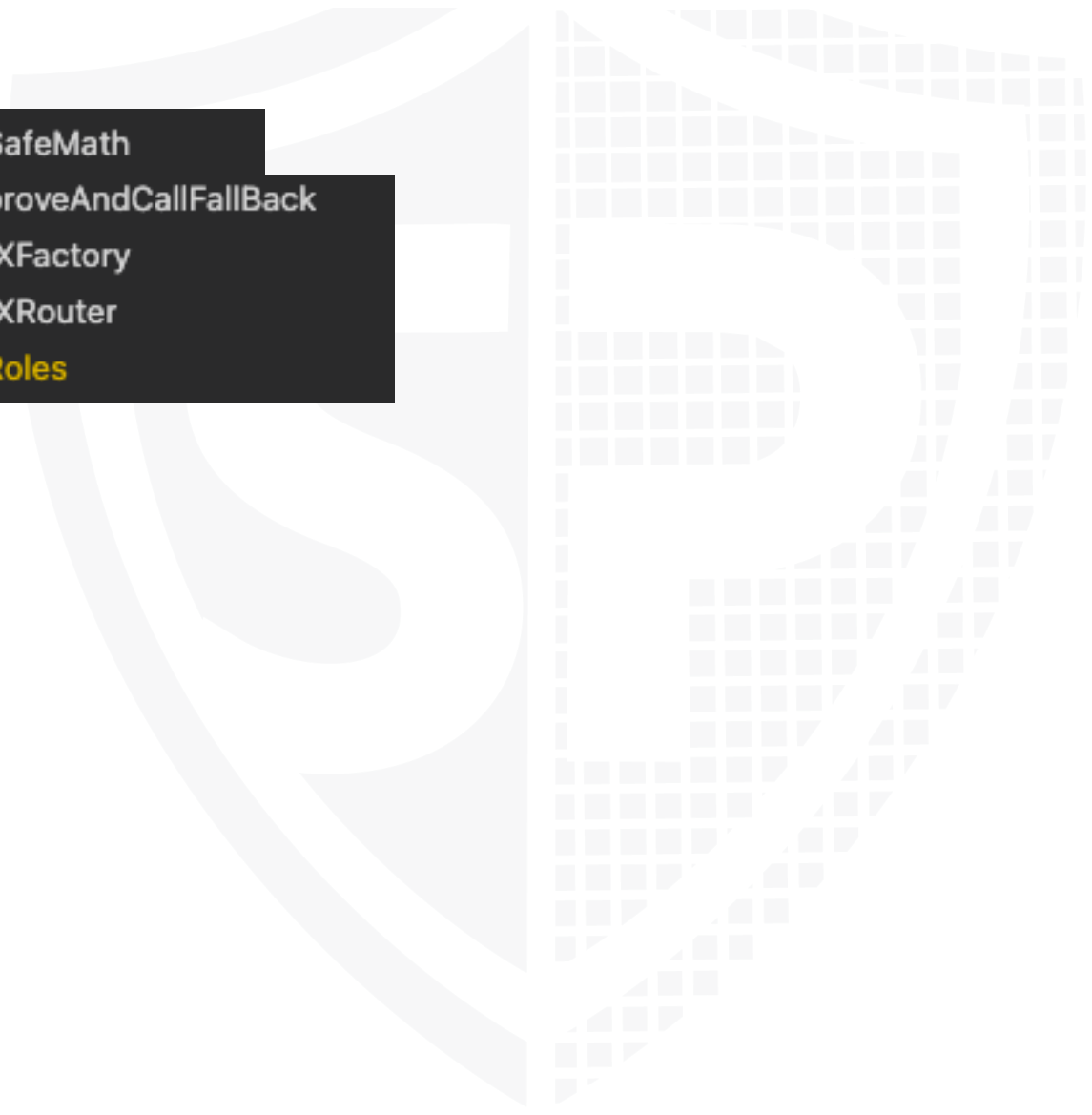
The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
 - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
 - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	1



SafeMath
ApproveAndCallFallback
IDEXFactory
IDEXRouter
Roles

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

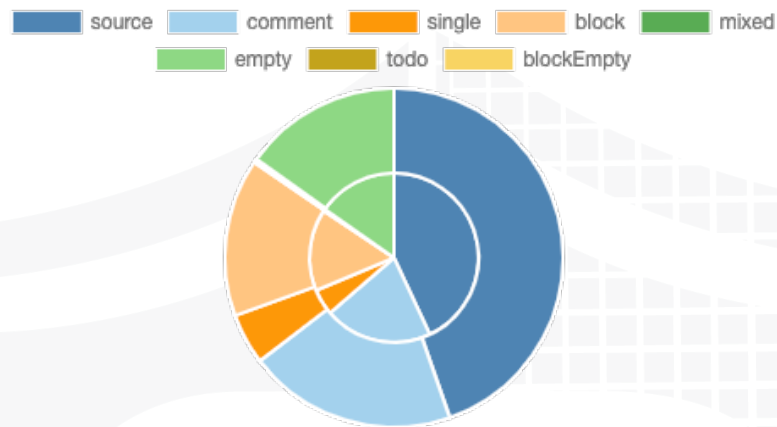
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

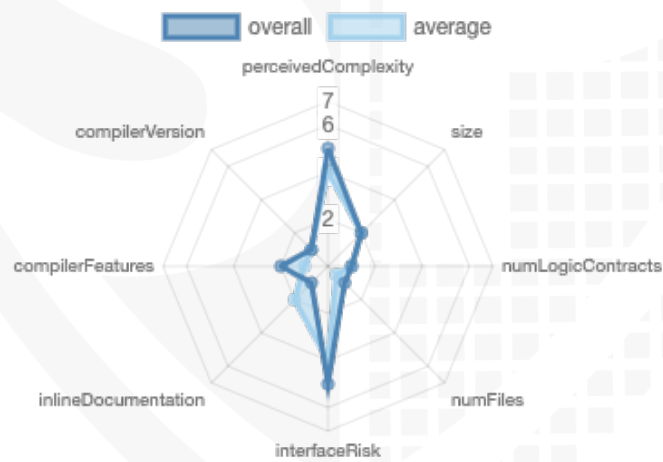
File Name	SHA-1 Hash
contracts/CairoToken.sol	aaea273e2bc1eded39e373c31edccdb05ee48df1
contracts/CairoMaximizer.sol	9198b82b51fb3b4a1ffbd130dde4986dfaacec4f

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	2	4	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.

Version	Public	Payable
1.0	79	3

Version	External	Internal	Private	Pure	View
1.0	49	86	0	11	29

State Variables

Version	Total	Public
1.0	45	18

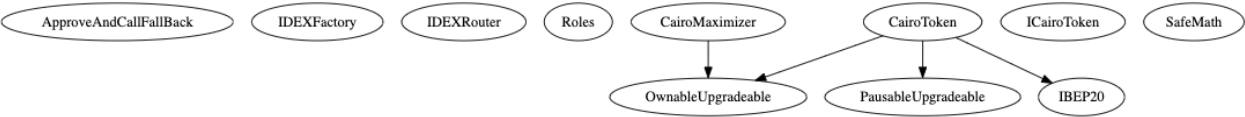
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.4		yes		

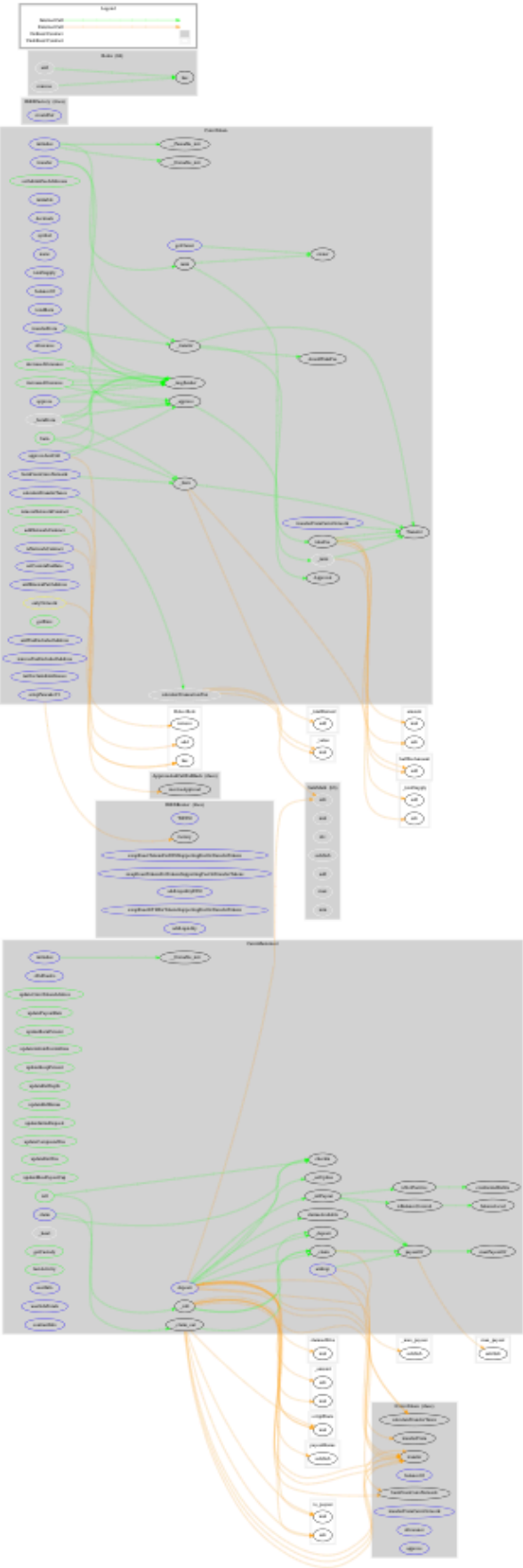
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes					

Inheritance Graph

v1.0



CallGraph v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Correct implementation of Token standard

ERC20				
Function	Description	Exist	Tested	Verified
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract v1.0

CairoToken

```
initialize
setAdminFeeAddresses
transfer
approve
approveAndCall
transferFrom
increaseAllowance
decreaseAllowance
burn
mint
removeNetworkContract
addNetworkContract
burnFromCairoNetwork
transferFromCairoNetwork
setCustomTaxRate
addKnownPairAddress
setupPancakeV1
addTaxExcludedAddress
removeTaxExcludedAddress
renounceOwnership
transferOwnership
```

CairoMaximizer

```
initialize
<Constructor> 💰
updateCairoTokenAddress
updatePayoutRate
updateBurnPercent
updateAdminFeeAddress
updateKeepPercent
updateRefDepth
updateRefBonus
updateInitialDeposit
updateCompoundTax
updateExitTax
updateMaxPayoutCap
checkin
deposit
claim
roll
airdrop
renounceOwnership
transferOwnership
```

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	1000000000		

Comments:

v1.0

- Owner mints new tokens in initialize function.



Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

- Owner can lock user funds by
 - Setting a custom tax rate for individual address to a high value
 - Owner cannot set `_hasCustomTax` variable per sender to false after setting it to true. That means, that the owner must set it manually back to 10 percent
- Tokens
 - can be burned by `msg.sender`
 - Can be burned by Cairo network without allowance from the address

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✓

Comments:

v1.0

- Deployer implements PausableUpgradeable in the contracts to use modifiers from the library. It is possible, that the owner can pause contracts in an upgraded version of token. Do your own research here



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

Modifiers and public functions

v1.0

Initialize	initializer
setAdminFeeAddresses	onlyOwner
transfer	
approve	whenNotPaused
approveAndCall	whenNotPaused
transferFrom	whenNotPaused
increaseAllowance	whenNotPaused
decreaseAllowance	whenNotPaused
burn	whenNotPaused
mint	onlyOwner
removeNetworkContract	onlyOwner
addNetworkContract	onlyOwner
burnFromCairoNetwork	onlyNetwork
transferFromCairoNetwork	onlyNetwork
setCustomTaxRate	onlyOwner
addKnownPairAddress	onlyOwner
setupPancakeV1	onlyOwner
addTaxExcludedAddress	onlyOwner
removeTaxExcludedAddress	onlyOwner

Initialize	initializer
<Constructor>	
updateCairoTokenAddress	onlyOwner
updatePayoutRate	onlyOwner
updateBurnPercent	onlyOwner
updateAdminFeeAddress	onlyOwner
updateKeepPercent	onlyOwner
updateRefDepth	onlyOwner
updateRefBonus	onlyOwner
updateInitialDeposit	onlyOwner
updateCompoundTax	onlyOwner
updateExitTax	onlyOwner
updateMaxPayoutCap	onlyOwner
checkin	
deposit	
claim	
roll	
airdrop	

Comments

- Deployer can set following state variables without any limitations
 - CairoToken
 - _customTaxRate
 - CairoMaximizer
 - payoutRate
 - maximizerBurnPercent
 - maximizerKeepAmount
 - ref_depth
 - ref_bonus

- minimumInitial
 - max_payout_cap
- Deployer can enable/disable following state variables
 - Token
 - _excluded
 - _isExcluded
- Deployer can set following addresses
 - CairoToken
 - router
 - pancakeV2BNBPair
 - pairs
 - networkContracts
 - allNetworkContracts
 - taxFeeSplit1
 - taxFeeSplit2
 - CairoMaximizer
 - cairoToken
 - adminFeeAddress
 - adminFeeAddress2
- Existing Modifiers
 - CairoToken
 - onlyNetwork
- CairoToken
 - Cairo networks can
 - transfer tokens without any allowance of the address
 - burn tokens without any allowance of the address
 - Owner can add any address to the networkContracts variable
 - All addresses are possible, not only contract addresses
 - Following functions cannot be called if contract is paused
 - Burn
 - Transfer
 - Approve
 - transferFrom
 - increaseAllowance
 - decreaseAllowance
- CairoMaximizer
 - custody.last_checkin has no functionality except of showing the time when the address interact with following functions
 - Deposit







- [Claim](#)
- [Roll](#)

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/CairoToken.sol	2	3	655	596	276	243	283	
	contracts/CairoMaximizer.sol	2	1	667	644	403	89	273	
	Totals	4	4	1322	1240	679	332	556	

Legend

Attribute	Description
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, ...)

Audit Results

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	CairoToken	A floating pragma is set	3	The current pragma Solidity directive is „^0.8.4“.
#2	CairoMaximizer	A floating pragma is set	3	The current pragma Solidity directive is „^0.8.4“.
#3	CairoToken	State variable visibility is not set	135, 136, 139, 140	It is best practice to set the visibility of state variables explicitly
#4	CairoToken	Local variables shadowing	490, 256	Rename the local variables that shadow another component
#5	CairoMaximizer	Missing Events Arithmetic	175, 170, 184, 145, 166, 162	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
-------	------	------	------	-------------

#1	CairoMaximizer	Functions that are not used	383	Remove unused functions
#2	CairoToken	Functions that are not used	504	Remove unused functions
#3	CairoToken	Error message is missing	103, 88, 89, 78, 79	Provide an error message for require statement
#4	CairoMaximizer	Error message is missing	179	Provide an error message for require statement
#5	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#6	CairoToken	Wrong comment	594	<i>"Transfer between two wallets the 10% on buy/sell"</i> is a wrong comment. The owner is able to set custom percentage for an individual address.

#9	CairoToken	Fee Calculation	592, See description	<p>Modify following variable:</p> <pre>uint256 halfFeeAmount = amount.mul(taxPercent).div(100).div(2);</pre> <p>To:</p> <pre>Uint256 fee = amount.mul(taxPercent).div(100); uint256 halfFee = fee.div(2); Uint256 otherHalfFee = fee.sub(halfFee);</pre> <p>The reason is for example the fees are an odd number (33) The half of 33 is normally 16,5 but solidity will cut of the 0,5 and instead it will be 16 per half. That means, that you are only transferring 32 instead of 33. So if you are calculate the half of 33 now, you will still get 16 because of solidity, but if you are subtraction 16 from 33 you will take the rest 1 with into your first/second taxFeeSplit.</p> <p>The result per taxSplit will be then:</p> <pre>taxFeeSplit1 = 16 taxFeeSplit1 = 17 Rest = 0</pre> <p>Instead of</p> <pre>taxFeeSplit1 = 16 taxFeeSplit1 = 16 Rest = 1</pre> <p>Same for</p> <ul style="list-style-type: none"> - adminHalfShare L237, L426, L395
----	------------	-----------------	----------------------	---

#10	CairoToken	Unnecessary library	See description	<p>You mustn't import SafeMath library into the contracts above pragma version 0.8.x anymore. It is already imported by default.</p> <p>You can use raw mathematical operations instead of library functions.</p>
#11	CairoMaximizer	Unused local variable	199, 526	<p>You are not using taxAmount in the deposit function. Remove the red marked text and leave the comma to solve this issue if you are not going to use that variable:</p> <pre>(uint256 realizedDeposit, uint256 taxAmount) = cairoToken.calculateTransferTaxes(_addr, _amount);</pre> <p>You can also replace the blue marked variable directly with _total_amount instead of setting _total_amount L200 to realizeDeposit</p> <p>L526:</p> <pre>(uint256 _gross_payout, uint256 _max_payout, uint256 _to_payout) = payoutOf(_addr);</pre>
#12	CairoMaximizer	Uninitialized variable	97, 105	minimumAmount will be 0, so there is no minimum deposit in deposit function
#13	CairoMaximizer	Variable with no effect	85	Remove variable if you are not going to use it
#14	CairoMaximizer	Contract that locks ether	-	You are not able to withdraw contract balance
#15	CairoMaximizer	Russian comment	477	<p>Translate russian comment into English</p> <ul style="list-style-type: none"> - Взгляды - совместимость - Пользователь в аплайне -

Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
CairoMaximizer	312	// require(users[_addr].upline != address(0) _addr == owner(), "No upline");
	628-630	// assert(b > 0); // Solidity automatically throws when dividing by 0 // uint256 c = a / b; // assert(a == b * c + a % b); // There is no case in which this doesn't hold

Recommendation

Remove the commented code, or address them properly.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

10. May 2022:

- Keep it in mind that the contracts are upgradeable
- Read whole report for more information

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the word "SolidProofed" in a white, handwritten-style script. The "P" is particularly large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProofed

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY