



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Fat Cat Killer

Audit

Security Assessment
07. May, 2022

For



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	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	21
Source Units in Scope	24
Critical issues	25
High issues	25
Medium issues	25
Low issues	25
Informational issues	26
Commented Code exist	27
Audit Comments	27
SWC Attacks	28

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	07. May 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://fatcatkiller.com/>

Telegram

<https://t.me/FatCatKillerPortal>

Twitter

https://twitter.com/Fat_Cat_Killer

Facebook

<https://www.facebook.com/FatCatKiller>

Instagram

<https://www.instagram.com/fatcatkillercoin/>

Github

Reddit

<https://www.reddit.com/r/FatCatKiller/>

Discord

<https://discord.gg/vwN7kn9H27>

Youtube

https://www.youtube.com/channel/UCTs0O5_wVvuvLjBrIT6ejsw

TikTok

<https://vm.tiktok.com/ZTdmEqj2J/>

LinkedIn

<https://www.linkedin.com/company/fatcatkiller>

Description

Fat Cat Killer is simply the best medium for merchant payment processing.

It supports all the ideological principles of crypto world: anonymity, speed, spanning across boarder, less governmental and financial institution control. While handling payments for luxury items and entertainment. Fat Cat Killer will provide unique benefits and access to its users and holders, at the same time directing a portion of its transaction fee to global charities, to give opportunity to those afflicted by income inequality.

Project Engagement

During the 5th of May 2022, **Fat Cat Killer 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
 - <https://github.com/TradingSession/fckcoin-contract/tree/tokenomics>
 - Commit: 1dc93b242a07f8ad6adf44c7087cbc72c9ab1376

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/access/Ownable.sol	8
@openzeppelin/contracts/security/Pausable.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	3
@openzeppelin/contracts/utils/math/Math.sol	4

Tested Contract Files

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

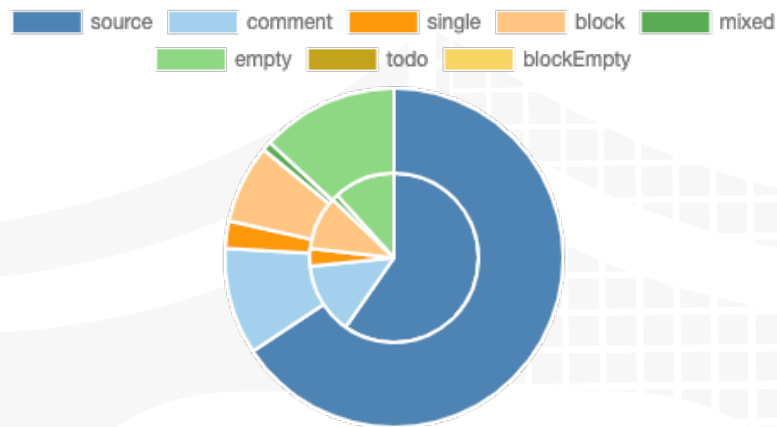
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

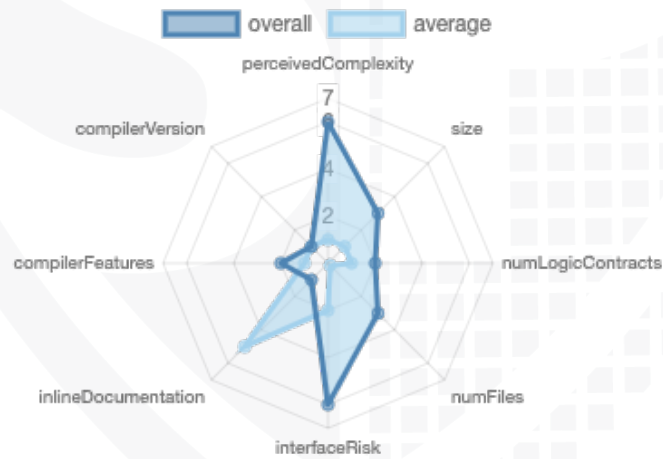
File Name	SHA-1 Hash
contracts/lottery/LotteryManager.sol	8c2a8ab92fb7367d397960407201af9c7c777d7c
contracts/control/IFCKController.sol	157a1fbdc1736e75af0bf1e7762e96fcb241f501
contracts/control/FCKController.sol	c1e28f32a1d246b70b824049952e6af6cdfb1cdc
contracts/wallets/TeamAndAdvisorsWallet.sol	f7bb267c73cce7c4aa46fc77b8399f3ee6349e9c
contracts/wallets/IPlatformReserveWallet.sol	ce5fa3324602465b180f6d5c15f008882353db16
contracts/wallets/IDistributionWallet.sol	7a5ca09099b52728d3ef32b7b5bf4e3da618ba42
contracts/wallets/MarketingReserveWallet.sol	6d47c5ca9d899cffe2457fff07e6c7e171d08683
contracts/wallets/LockupWallet.sol	e6f641af52670183366698f5b13c9c581a8501e3
contracts/wallets/PlatformReserveWallet.sol	9ddd1e2cb07b49afbcfdfe72ad6d2956b4df8f73
contracts/governance/IVoting.sol	bc0423d0b03412bcfe9974cecb8953b5449403f2
contracts/governance/Voting.sol	819e87da612a86c8865c59086bdd3b86a3afee23
contracts/strategies/ManualTokenomicsStrategy.sol	c2706f6770a77078c99b1ebb40fda1905ae38ac3
contracts/strategies/BaseTokenomicsStrategy.sol	88b3d1d35f2cf3e85b5a988548b755df417c6629
contracts/strategies/ITokenomicsStrategy.sol	bee920f4247efbc485623bbd1be605f6779f534
contracts/strategies/AutomaticTokenomicsStrategy.sol	9b9b217153a5436f7bc5fc13417fb9a28db96d9d
contracts/pancake/IPancakePair.sol	e3894616e0954da95f3d3daab00b8b91d02f500a
contracts/pancake/IPancakeRouter02.sol	b01261a9f0fda3a7a23cfec8198480490cb7d78d
contracts/pancake/IPancakeRouter01.sol	22fef0faf85d7c7a4c2ae511ac3d1a61a6cb5efb
contracts/pancake/IPancakeFactory.sol	c1c6067588ad46971ed00f7fa5ce2431889326d0
contracts/token/FCKToken.sol	fd1def2b0bcd3834dab68a94bd0039990e34a05b
contracts/token/TokenomicsToken.sol	6d1bef558a25a8663e237ebb6a255085b9fd284e
contracts/token/ITokenomicsToken.sol	8bf77917b3ccce9c44529b4840939c386fb88e87
contracts/token/ERC20.sol	7bd5b8d59719bfdd14d14271cbf593b011a24659
contracts/token/IFCKToken.sol	a4b4e094255e90c0bfc46fc4e44aea1325b7e6c0

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	13	0	11	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	214	6

Version	External	Internal	Private	Pure	View
1.0	201	160	2	12	85

State Variables

Version	Total	Public
1.0	76	4

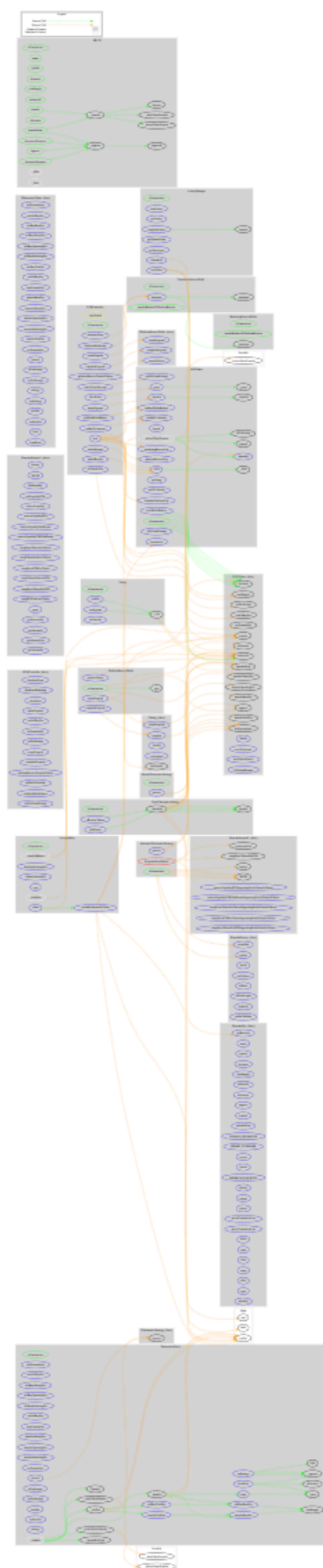
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	>=0.8.0 <0.9.0 >=0.5.0 >=0.6.2 >=0.5.16 ^0.8.0		yes		

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

Inheritance Graph 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

start
tokenPause
tokenUnpause
distributeTeam
distributeMarketing
setSellBuyFee
setTransferFee
setFeeExempt
createProposal
completeProposal
platformReserveTransferTokens
setMaxTxAmount
setMaxWalletBalance
setIsTxLimitExempt

process

launch
mint
pause
unpause
setVoting
setMaxTxAmount
setMaxWalletBalance
setIsTxLimitExempt

createProposal
voteFor
voteAgainst
complete

setSellBuyFee
setTransferFee
process
setFeeExempt
setDexPair
setStrategy
burn
burnFrom

startLottery
buyTicket
completeLottery
claimHold

transfer
approve
transferFrom
increaseAllowance
decreaseAllowance

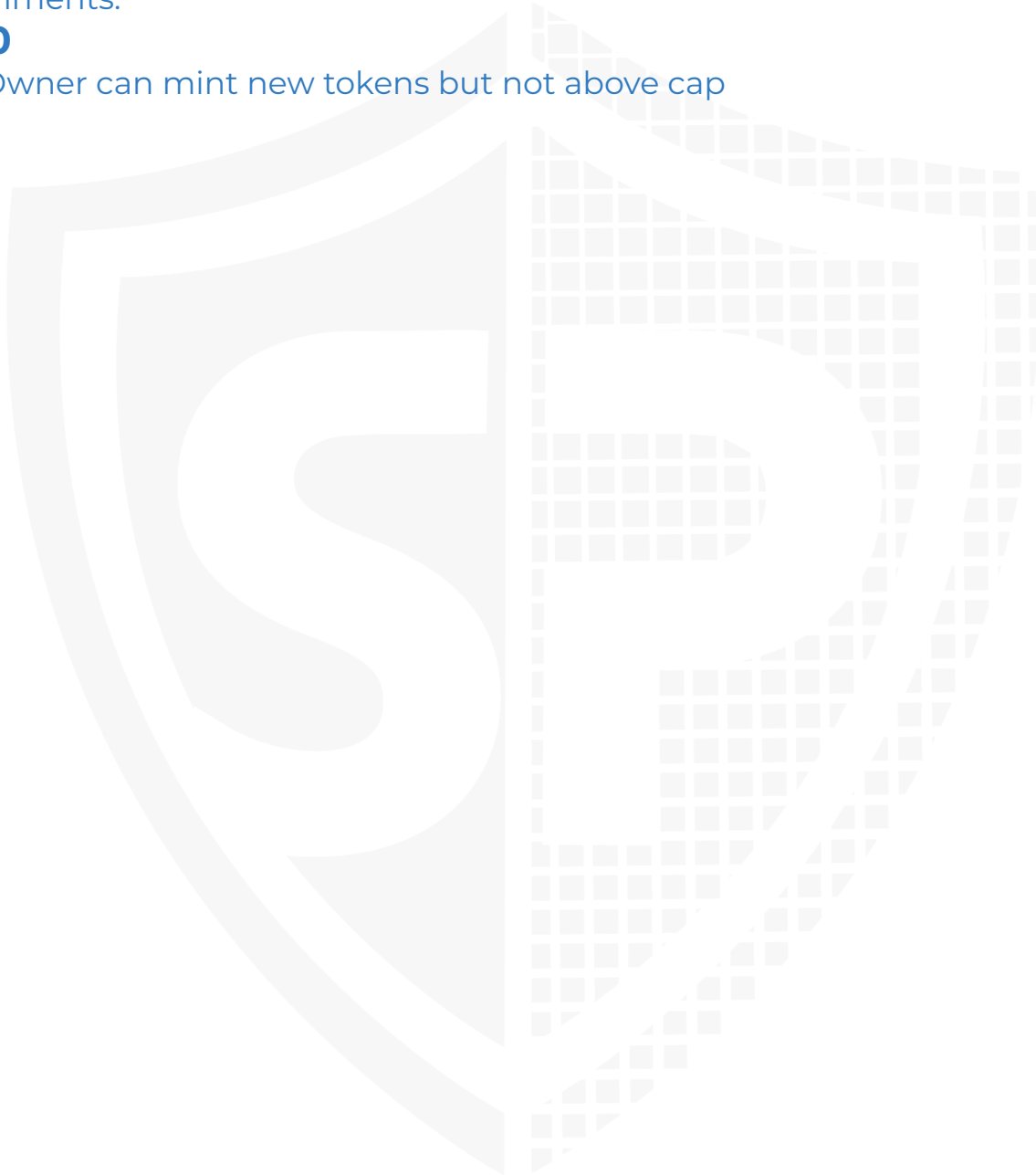
Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

- Owner can mint new tokens but not above cap



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 max tx amount to 0
 - Setting max wallet amount to 0
- Tokens
 - will be burned while tx
 - can be burned by msg.sender

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Owner can pause contract



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

- launch
- mint
 - onlyOwner
- pause
 - onlyOwner
- unpause
 - onlyOwner
- setVoting
 - onlyOwner
- setMaxTxAmount
 - onlyOwner
- setMaxWalletBalance
 - onlyOwner
- setIsTxLimitExempt
 - onlyOwner

- createProposal
 - onlyOwner
- voteFor
- voteAgainst
- complete
 - onlyOwner

- startLottery
 - onlyOwner
- buyTicket
- completeLottery
 - onlyOwner
- claimHold

process

- start
- tokenPause
 - onlyOwner
- tokenUnpause
 - onlyOwner
- distributeTeam
 - onlyOwner
 - onlyStarted
- distributeMarketing
 - onlyOwner
 - onlyStarted
- setSellBuyFee
 - onlyOwner
- setTransferFee
 - onlyOwner
- setFeeExempt
 - onlyOwner
- createProposal
 - onlyOwner
- completeProposal
 - onlyOwner
- platformReserveTransferTokens
 - onlyOwner
- setMaxTxAmount
 - onlyOwner
- setMaxWalletBalance
 - onlyOwner
- setIsTxLimitExempt
 - onlyOwner

- setSellBuyFee
 - onlyOwner
- setTransferFee
 - onlyOwner
- process
 - onlyOwner
- setFeeExempt
- setDexPair
 - onlyOwner
- setStrategy
 - onlyOwner
- burn
- burnFrom

claim

Information: Not listed functions/modifiers etc. are from libraries

Comments

- Deployer can set following state variables without any limitations
 - _maxTxAmount
 - _maxWalletBalance
- Deployer can enable/disable following state variables
 - _paused
 - _isFeeExempt
 - _isTxLimitExempt
 - _voting
- Deployer can set following addresses


































- `_dexPair`
- `_strategy`
- *Existing Modifiers*
 - `onlyStarted`

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/lottery/LotteryManager.sol	1	————	157	129	113	1	45	
	contracts/control/IFCKController.sol	————	1	48	6	3	6	27	————
	contracts/control/FCKController.sol	1	————	154	116	93	4	81	————
	contracts/wallets/TeamAndAdvisorsWallet.sol	1	————	23	19	14	1	16	————
	contracts/wallets/IPlatformReserveWallet.sol	————	1	14	5	3	1	7	————
	contracts/wallets/IDistributionWallet.sol	————	1	6	5	3	1	3	————
	contracts/wallets/MarketingReserveWallet.sol	1	————	23	19	14	1	15	————
	contracts/wallets/LockupWallet.sol	1	————	114	106	81	5	40	
	contracts/wallets/PlatformReserveWallet.sol	1	————	47	40	33	2	26	
	contracts/governance/IVoting.sol	————	1	18	5	3	1	11	————
	contracts/governance/Voting.sol	1	————	137	133	120	1	39	————
	contracts/strategies/ManualTokenomicsStrategy.sol	1	————	40	40	35	1	15	————
	contracts/strategies/BaseTokenomicsStrategy.sol	1	————	70	70	60	1	35	
	contracts/strategies/ITokenomicsStrategy.sol	————	1	6	5	3	1	3	————
	contracts/strategies/AutomaticTokenomicsStrategy.sol	1	————	83	80	64	13	39	————
	contracts/pancake/IPancakePair.sol	————	1	111	12	9	1	55	————
	contracts/pancake/IPancakeRouter02.sol	————	1	51	7	4	1	16	
	contracts/pancake/IPancakeRouter01.sol	————	1	161	5	3	1	48	
	contracts/pancake/IPancakeFactory.sol	————	1	32	12	9	1	17	————
	contracts/token/FCKToken.sol	1	————	145	128	102	7	86	————
	contracts/token/TokenomicsToken.sol	1	————	311	269	223	12	152	
	contracts/token/ITokenomicsToken.sol	————	1	68	9	6	1	53	————
	contracts/token/ERC20.sol	1	————	386	336	109	194	58	
	contracts/token/IFCKToken.sol	————	1	44	9	6	1	33	————
	Totals	13	11	2249	1565	1113	259	920	

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	All	A floating pragma is set	See description	The current pragma Solidity directives are floating pragmas.
#2	AutomaticTokenomicsStrategy	Missing Zero Address Validation (missing-zero-check)	30-33	Check that the address is not zero
#3	BaseTokenomicsStrategy	Missing Zero Address Validation (missing-zero-check)	14-17	Check that the address is not zero
#4	ManualTokenomicsStrategy	Missing Zero Address Validation (missing-zero-check)	19	Check that the address is not zero
#5	TokenomicsToken	Missing Zero Address Validation (missing-zero-check)	156	Check that the address is not zero

#6	FCKToken	Missing Events Arithmetic	114, 126	Emit an event for critical parameter changes
#7	Lottery Manager	Weak PRNG	131	We recommend to use an external service like Chainlink VRF https://docs.chain.link/docs/chainlink-vrf/

Informational issues

Issue	File	Type	Line	Description
#1	AutomaticTokenomicsStrategy	Functions that are not used	48	Remove unused functions
#2	Lockup Wallet	Functions that are not used	107	Remove unused functions
#3	AutomaticTokenomicsStrategy	Unused state variables	19, 20, 21	Remove unused state variables
#4	Lockup Wallet	Unused state variables	11	Remove unused state variables
#5	Lockup Wallet	Error message is missing	100	Provide an error message for require statement
#6	BaseTokenomicsStrategy	Low level calls	55, 59, 63	Check low level success status after execution

Commented Code exist

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

File	Line	Comment
AutomaticTokenomicsStrategy	75-81	<pre>// charityAmount = (amount * (_token.shouldBurnFee() ? 10 : 33)) / 100; // operationalAmount = amount - (amountToBurn + charityAmount); // if (_token.shouldBurnFee()) { // _token.burnFrom(address(this), amountToBurn); // } // SwapAndSendTokens(charityAmount, _charityWallet); // SwapAndSendTokens(operationalAmount, _operationalWallet);</pre>

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.

07. May 2022:

- 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	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	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 words "SolidProof" in a white, handwritten-style script. The "P" is 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.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

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

MADE IN GERMANY