SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

22. November, 2021

For

KAWAII SWAP

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 Unicrypt, 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	22. November 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://www.kawaiiswap.finance/

Telegram
https://t.me/KawaiiSwapAnn
https://t.me/kawaiiswap

Twitter
https://twitter.com/kawaiiswap

Discord
https://discord.gg/rhkHuSMzTR

Description

We enrich traditional yield farming experience with gamification features. By connecting gaming world with farming platform through Layers we provide constant utility to the native token therefore ensuring continuous growth of the project.

KawaiiSwap users are able to win tokens and NFTs in games run on the platform or complete quests to gain APR boosts. NFTs can be traded on the marketplace or used for in-game activities. Governance token CALCIFIRE holders are able to become shareholders and receive dividends just by holding tokens in the wallet as well as to participate in decision-making process.

KawaiiSwap project is backed by the team of professional developers and belongs to "Brainstorm Digital" Ltd company. Our vision is to extend user interaction with the platform way beyond yield farming platform by creating fictional world with play to earn model that combines traditional RPG experience with NFT ownership.

Project Engagement

During the 19th of November 2021, **Calcifire 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

KAWAII SWAP

Contract Link

v1.0

MasterChef: https://bscscan.com/address/0x1d0b3f48e15636caa51e41a20a8a82cdedc982ed#code

Calcifire Token: https://bscscan.com/address/0x822866238d113a2fe77d5b5b5a0cdb895df90637#code

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 aspossible.
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:

Calcifire

Address.sol
CalcifireToken.sol
Context.sol
IBEP20.sol
IUniswapV2Factory.sol
IUniswapV2Pair.sol
IUniswapV2Router01.sol
IUniswapV2Router02.sol

Ownable.sol

MasterChef

Address.sol
CalcifireToken.sol
Context.sol

IBEP20.sol
ICalcifireReferral.sol
IERC20.sol
IUniswapV2Factory.sol
IUniswapV2Pair.sol
IUniswapV2Router01.sol
IUniswapV2Router02.sol
Migrations.sol
Ownable.sol
ReentrancyGuard.sol
SafeBEP20.sol
SafeMath.sol

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
Calcifire

File Name	SHA-1 Hash
contracts/IBEP20.sol	8e83c72a70de313ccbb600d3d9c1297df8d6fef0
contracts/IUniswapV2Pair.sol	517a6c11a937212f62e409114817e7b3decd2451
contracts/Context.sol	02ebe0e93c5d1da25b91ba7f4cfb990a949263f8
contracts/IUniswapV2Factory.sol	ec741c917da6d8a7adf874d00d5dbf7f220d25ed
contracts/CalcifireToken.sol	885f502b2a46d7987f7f67b6653693408726a12d
contracts/Address.sol	93871f4002f27f5e67e6ad0c243d868d321d9234
contracts/SafeMath.sol	16904ca20d27ddfca0969cc322c39d159d33aa57
contracts/Ownable.sol	276129ff22713a5e32a785d4b72eea81e72912b2
contracts/IUniswapV2Router02.sol	8665c0ca5a56e579135c5c24af9502a188627e6c
contracts/IUniswapV2Router01.sol	ed8695e6d43176bac9ba7517f4952886009d1dc8

MasterChef

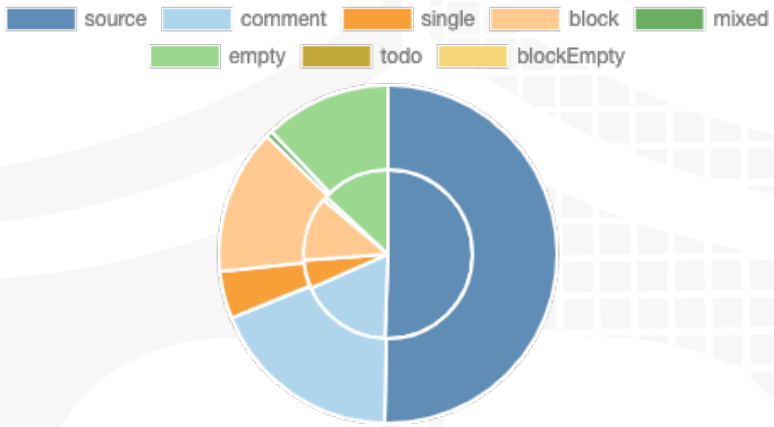
File Name	SHA-1 Hash
contracts/IBEP20.sol	8e83c72a70de313ccbb600d3d9c1297df8d6fef0
contracts/HowlsCastle.sol	73dc7c89f603882e93f9ccd1af8904824ee2001c
contracts/IUniswapV2Pair.sol	517a6c11a937212f62e409114817e7b3decd2451
contracts/Context.sol	02ebe0e93c5d1da25b91ba7f4cfb990a949263f8
contracts/SafeBEP20.sol	96bc8a79b9bd44b8d86c0a7dc9d5560929463755
contracts/IUniswapV2Factory.sol	ec741c917da6d8a7adf874d00d5dbf7f220d25ed
contracts/CalcifireToken.sol	885f502b2a46d7987f7f67b6653693408726a12d
contracts/ICalcifireReferral.sol	97e2ca12a87900a012fe5e5c0dc0a24319b1e496
contracts/Address.sol	66db1de364ee244b292cf4cc5e63385e8f6b9420
contracts/SafeMath.sol	16904ca20d27ddfca0969cc322c39d159d33aa57
contracts/Ownable.sol	276129ff22713a5e32a785d4b72eea81e72912b2
contracts/IUniswapV2Router02.sol	8665c0ca5a56e579135c5c24af9502a188627e6c
contracts/IUniswapV2Router01.sol	ed8695e6d43176bac9ba7517f4952886009d1dc8
contracts/ReentrancyGuard.sol	4ac50679fc09a3ea3d6dd545e187063e9a05be2c
contracts/IERC20.sol	3ed225ee21131d8e3b6fee8fd5265ab0292d7328

Metrics

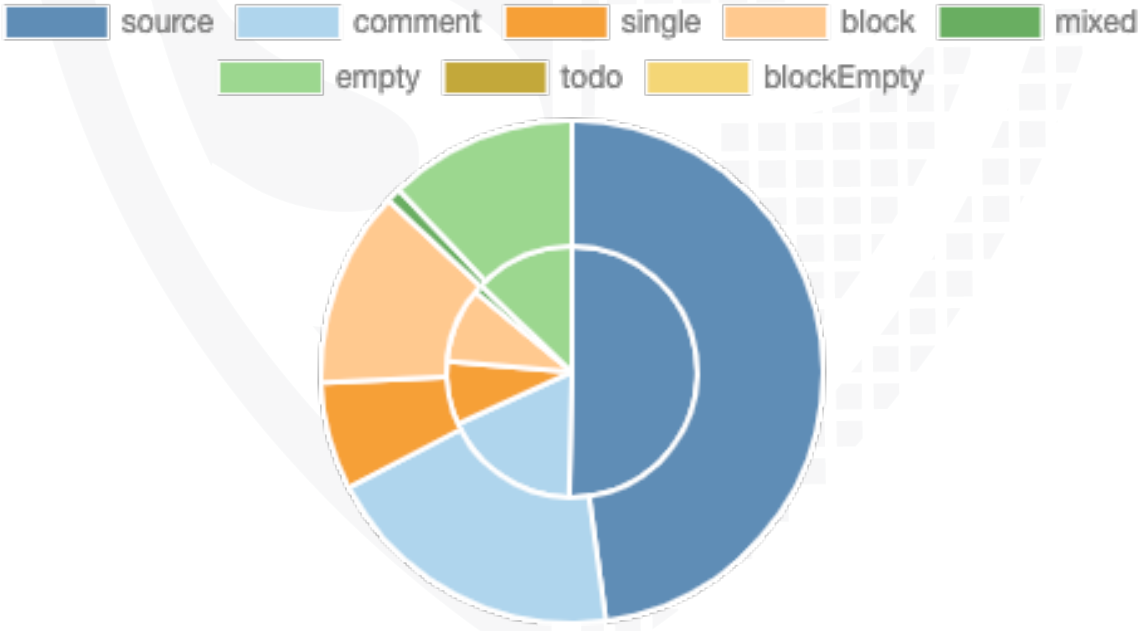
Source Lines

v1.0

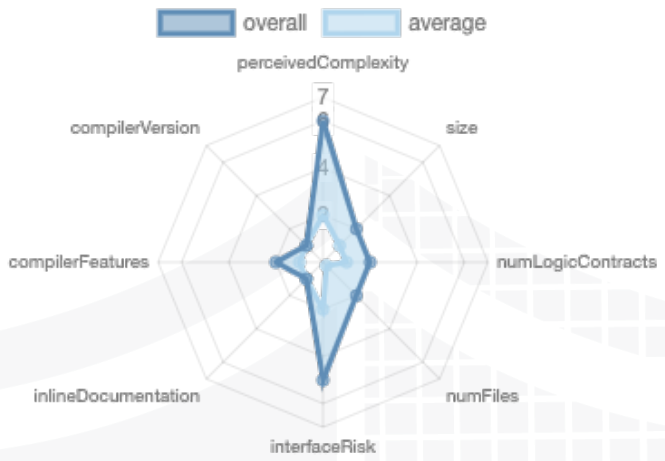
Calcifire



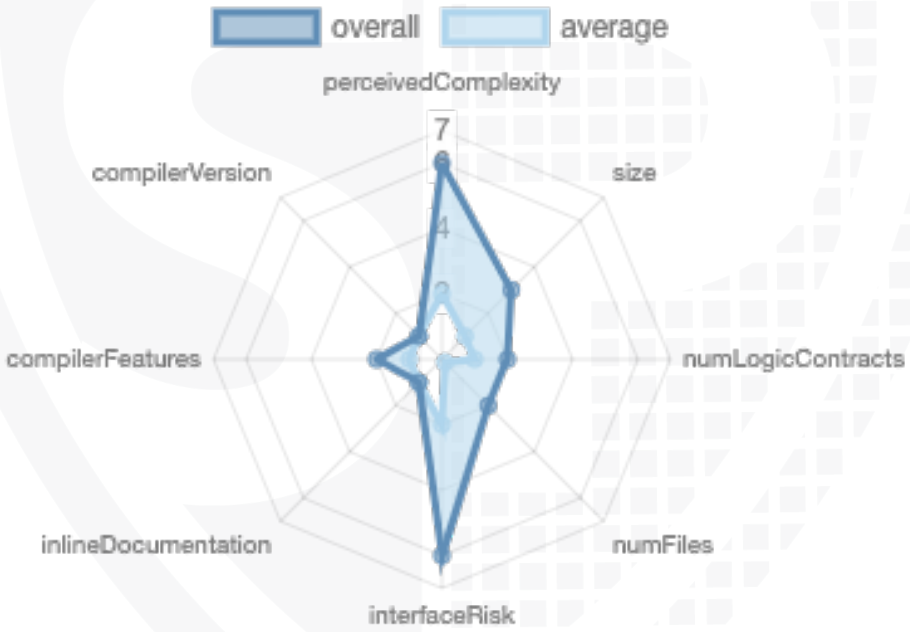
MasterChef



Risk Level
v1.0



MasterChef



Capabilities

Components

File	Version	Contracts	Libraries	Interfaces	Abstract
Calcifire	1.0	2	2	5	1
MasterChef	1.0	3	3	7	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

File	Version	Public	Payable
Calcifire	1.0	135	5
MasterChef	1.0	180	5

File	Version	External	Internal	Private	Pure	View
Calcifire	1.0	92	123	28	21	60
MasterChef	1.0	106	177	29	23	78

State Variables

File	Version	Total	Public
Calcifire	1.0	58	28
MasterChef	1.0	94	59

Capabilities

File	Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
Calcifire	1.0	>=0.6.4 0.6.12 >=0.6.0 <0.8.0 >=0.6.2 <0.8.0		yes	yes (3 asm blocks)	
MasterChef	1.0	>=0.6.4 0.6.12 >=0.6.0 <0.8.0 >=0.6.2 <0.8.0		yes	yes (3 asm blocks)	

File	Version	Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
Calcifire	1.0				yes	yes	
MasterChef	1.0	yes		yes	yes	yes	

Scope of Work

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).

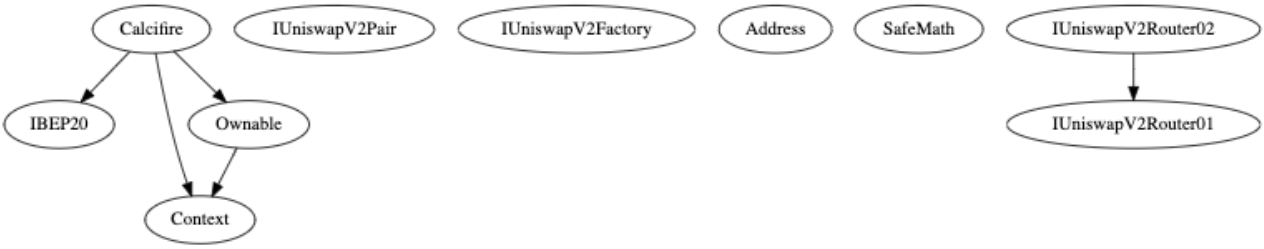
We will verify the following claims:

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)

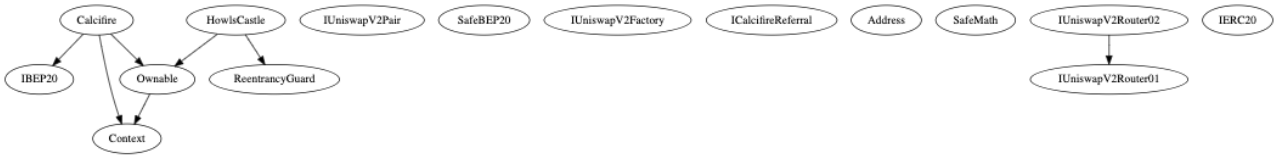
Inheritance Graph

v1.0

Calcifire



MasterChef



Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

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 Calcifire

1. approve
2. calculateReflection
3. decreaseAllowance
4. delegate
5. delegateBySig
6. distribute
7. excludeFromFee
8. excludeFromReward
9. includeInFee
10. includeInReward
11. increaseAllowance
12. lock
13. mint
14. openTrading
15. renounceOwnership
16. setAutoSellForCommunity
17. setAutoSellForTreasury
18. setAutomatedMarketMakerPair
19. setBurnFeePercent
20. setBurnToBurnAddress
21. setCommunityAddress
22. setCommunityFeePercent
23. setDeflationary
24. setExcludedFromAntiWhale
25. setLiquidityFeePercent
26. setLiquidityTaxAddress
27. setNumTokensSellToAddToLiquidity
28. setSellFeePercents
29. setSwapAndLiquifyEnabled
30. setTaxFeePercent
31. setTreasuryAddress
32. setTreasuryFeePercent
33. setWallet
34. transfer
35. transferFrom
36. transferOperator
37. transferOwnership
38. triggerSwapAndLiquify
39. unlock
40. updateAutoAddLiquidityRouter
41. updateMaxTransferAmountRate

MasterChef

1. add →
2. addUserBoostByOperator →
3. addUserPoolBoostByOperator →
4. deposit →
5. emergencyWithdraw →
6. lock →
7. massUpdatePools →
8. renounceOwnership →
9. set →
10. setBoostAmounts →
11. setBoostLimitsEnabled →
12. setDevAddress →
13. setFeeAddress →
14. setPoolBoost →
15. setPoolBoostUSDLimits →
16. setReferralCommissionRate →
17. setReferralContract →
18. setStartRewardBlock →
19. setTokenUSDPair →
20. transferOwnership →
21. unlock →
22. updateEmissionHalving →
23. updateEmissionRate →
24. updateOperator →
25. updatePool →
26. withdraw →

Deployer cannot mint any new tokens

File	Name	Exist	Tested	Verified
Calcifire	Deployer cannot mint	✓	✓	✗
MasterChef	Deployer cannot mint	-	-	-

Max / Total Supply: 3.000.000

Comments:

v1.0

- Calcifire
  - Deployer can mint until isDeflationary is set to true
- MasterChef
  - Calcifire.mint ist called in
    - updatePool
    - payOrLockupPendingCALCIFIRE
    - payReferralCommision

Deployer cannot burn or lock user funds

File	Name	Exist	Tested	Verified
Calcifire	Deployer cannot lock	✓	✓	✓
	Deployer cannot burn	✓	✓	✓
MasterChef	Deployer cannot lock	✓	✓	✓
	Deployer cannot burn	-	-	-

Comments:
v1.0
.

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
Calcifire	Deployer cannot pause	✓	✓	✓
MasterChef	Deployer cannot pause	✓	✓	✓

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

OnlyOwner functions

Calcifire

mint
setDeflationary
transferOperator
setExcludedFromAntiWhale
updateMaxTransferAmountRate
updateAutoAddLiquidityRouter
setAutomatedMarketMakerPair
distribute

excludeFromReward
includeInReward
excludeFromFee
includeInFee
setSellFeePercents
openTrading
setTaxFeePercent
setLiquidityFeePercent
setCommunityFeePercent
setTreasuryFeePercent
setBurnFeePercent
setTreasuryAddress
setCommunityAddress
setLiquidityTaxAddress
setSwapAndLiquifyEnabled
setNumTokensSellToAddToLiquidity
setAutoSellForTreasury
setAutoSellForCommunity
setBurnToBurnAddress
triggerSwapAndLiquify

Comments:

- updateMaxTransferAmountRate
  - maxTransferAmountRate can be set only between 50 and 1000
- setSellFeePercents
  - _sellTaxFee + _sellLiquidityFee + _sellBurnFee + _sellTreasuryFee + _sellCommunityFee can be set only up to 20

- setTaxFeePercent
  - taxFee + _liquidityFee + _communityFee + _treasuryFee + _burnFee can be set only up tp 10

MasterChef

- onlyOperator

```
setTokenUSDPair
setBoostLimitsEnabled
setPoolBoostUSDLimits
```

```
addUserBoostByOperator
addUserPoolBoostByOperator
```

- onlyOwner

```
add
set
```

```
setBoostAmounts
setPoolBoost
```

```
updateOperator
updateEmissionRate
updateEmissionHalving
setReferralContract
setReferralCommissionRate
setStartRewardBlock
```

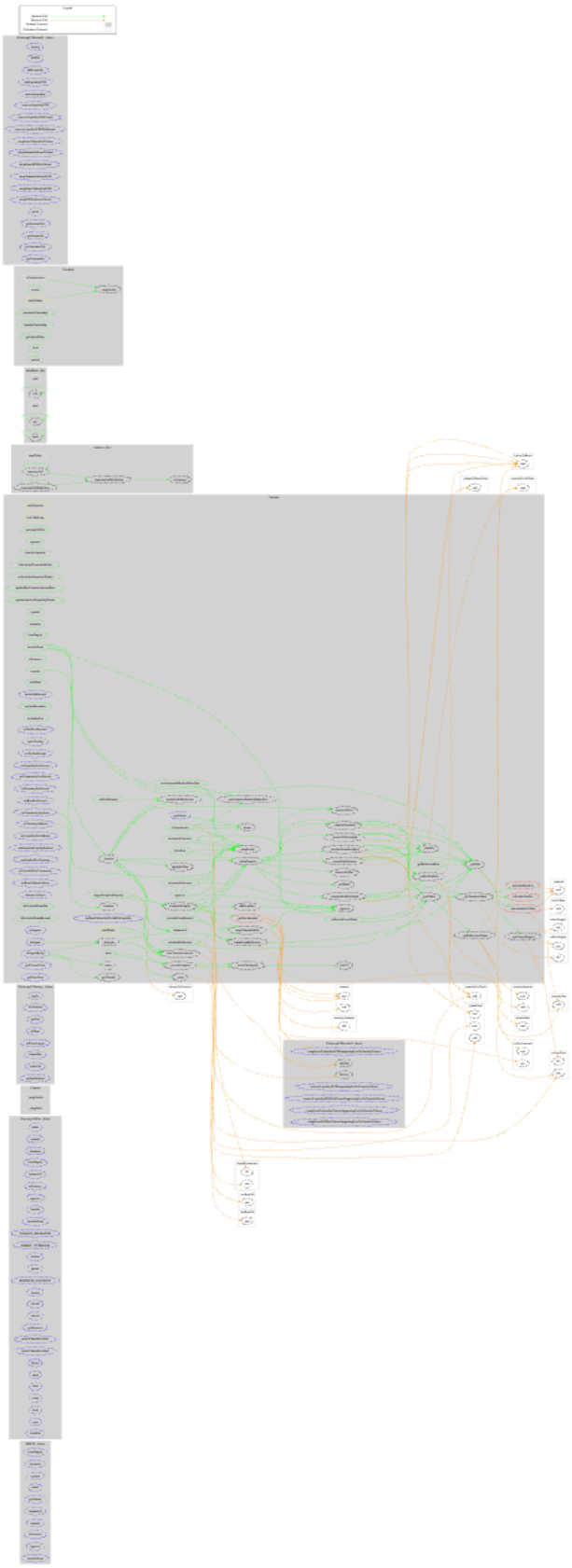
- onlyDev
  - setDevAddress
- onlyFeeAddress
  - setFeeAddress

Comments:

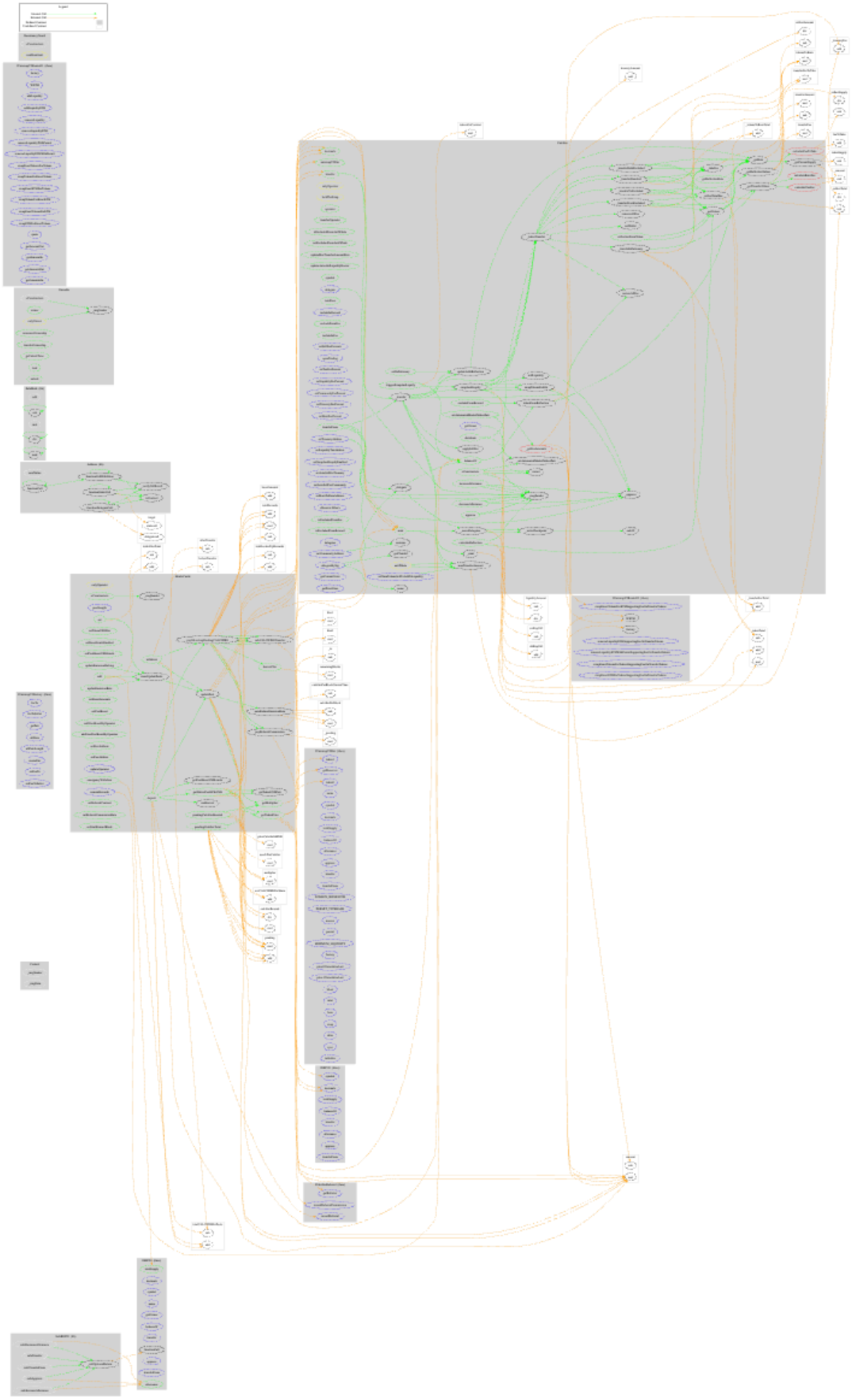
- setBoostAmounts
  - Following state variables can be set without any limitations
    - maxPoolBoostAmount
    - maxUserBoostAmount
    - userPoolBoostAmount
- updateEmissionHalving
  - Following state variables can be set without any limitations
    - calcifireHalvingInterval
    - emissionRateDecreasePerBlock
    - targetCalcifirePerBlock
- setStartRewardBlock

- Owner can set startBlock without limitations

CallGraph
Calcifire



MasterChef



Source Units in Scope

v1.0

Calcifire

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/IBEP20.sol	———	1	93	22	17	66	21	———
🔍	contracts/IUniswapV2Pair.sol	———	1	54	9	5	1	55	———
🎨	contracts/Context.sol	1	———	24	24	10	12	1	🔆
🔍	contracts/IUniswapV2Factory.sol	———	1	19	8	4	1	17	———
📝	contracts/CalcifireToken.sol	1	———	1110	1077	777	112	600	💻💰🧱💉
📚	contracts/Address.sol	1	———	141	126	55	87	37	💻🔆
📚	contracts/SafeMath.sol	1	———	159	159	39	106	10	🔆
📝	contracts/Ownable.sol	1	———	90	90	44	35	38	———
🔍	contracts/IUniswapV2Router02.sol	———	1	46	8	4	1	16	💰
🔍	contracts/IUniswapV2Router01.sol	———	1	97	6	3	1	48	💰
📝📚🔍🎨	**Totals**	**5**	**5**	**1833**	**1529**	**958**	**422**	**843**	💻💰🧱💉🔆

MasterChef

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/IBEP20.sol	———	1	93	22	17	66	21	———
📝	contracts/HowlsCastle.sol	1	———	692	692	481	115	419	🔷🧱
🔍	contracts/IUniswapV2Pair.sol	———	1	54	9	5	1	55	———
🎨	contracts/Context.sol	1	———	24	24	10	12	1	🔆
📚	contracts/SafeBEP20.sol	1	———	75	74	33	32	25	———
🔍	contracts/IUniswapV2Factory.sol	———	1	19	8	4	1	17	———
📝	contracts/CalcifireToken.sol	1	———	1110	1077	777	112	600	💻💰🧱💉
🔍	contracts/ICalcifireReferral.sol	———	1	20	9	3	10	7	———
📚	contracts/Address.sol	1	———	189	169	78	113	47	💻👥
📚	contracts/SafeMath.sol	1	———	159	159	39	106	10	🔆
📝	contracts/Ownable.sol	1	———	90	90	44	35	38	———
🔍	contracts/IUniswapV2Router02.sol	———	1	46	8	4	1	16	💰
🔍	contracts/IUniswapV2Router01.sol	———	1	97	6	3	1	48	💰
🎨	contracts/ReentrancyGuard.sol	1	———	62	62	15	38	5	———
🔍	contracts/IERC20.sol	———	1	84	21	17	57	17	———
📝📚🔍🎨	**Totals**	**8**	**7**	**2814**	**2430**	**1530**	**700**	**1326**	💻💰🔷👥🧱💉🔆

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 found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	Calcifire, MasterChef	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Calcifire	Missing Zero Address Validation (missing-zero-check)	129, 462	Check that the address is not zero
#3	MasterChef	Missing Zero Address Validation (missing-zero-check)	147, 148, 149	Check that the address is not zero
#4	Calcifire	State variable visibility is not set	70	It is best practice to set the visibility of state variables explicitly
#5	Calcifire	Missing Events Arithmetic	447, 437, 432, 471, 413, 427, 442	Emit an event for critical parameter changes
#6	Calcifire	Usage of equality comparison instead of assignment	681	This equality comparison doesn't have any effect. If you want to assign a variable, please use only one equal sign instead of ==

#7	Calcifire	Use of raw math arithmetics	1036	We recommend to use safe math functions instead of using raw math arithmetics

Informational issues

Issue	File	Type	Line	Description
#1	Calcifire	State variables that could be declared constant (constable-states)	65, 35, 33, 34	Add the `constant` attributes to state variables that never change

Commented Code exist

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

Line	Comment
Calcifire, 382	// require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap router.');

Recommendation

Remove the commented code, or address them properly.

Audit Comments

22. November 2021:

- Line 84, Calcifire, _operator wrong comment
  - _operator is not only for update the MaxTransferAmountRate and tax rate, it is used as an onlyOwner modifier (see onlyOwner functions, Calcifire)
- calcifireReferral was not provided to solidproof
  - We recommend you to do your own research about that contract

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	NOT PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY