SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 19. November, 2021

Audit

Security Assessment
04. December, 2021

For

ZONOSWAP

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	19. November 2021	• Layout project • Automated- /Manual-Security Testing • Summary
2.0	04. December 2021	• Audit

Network
Binance Smart Chain (BEP20)

Website
https://zonoswap.com/

Telegram
https://t.me/zonoswap

Twitter
https://twitter.com/ZonoSwap

Facebook
https://www.facebook.com/Zonoswap

Github
https://github.com/Zonoswap/

Description

A decentralized exchange (DEX) is a cryptocurrency exchange that operates without a central authority, allowing users to transact peer-to-peer from wallet-to-wallet whilst maintaining complete control of their assets. DEXs reduce the risk of price manipulation, as well as hacking and theft, because crypto assets are never in the custody of the exchange itself.

Project Engagement

During the 15th of November 2021, **ZonoSwap 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

ZONOSWAP

Contract Link

v1.0

MasterChef: https://bscscan.com/address/0xaff2b0031aa7a5c19c9cba9589817a704234d492#code

TokenAddress: https://bscscan.com/address/0x8d08dcb48d59216ae5d0515aa6622c9beb42b76b#code

RouterAddress: https://bscscan.com/address/0x1a547b5ce91ba3b65305e7979a12c8bc8a3d4962

FactoryAddress: https://bscscan.com/address/0xd43f10afc9b2dcb7b85e612734958e12124ac84f#code

V2.0

TokenAddress: https://bscscan.com/address/0xAc97796B45F9627e16da9C93e608579ceEb410a4#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.

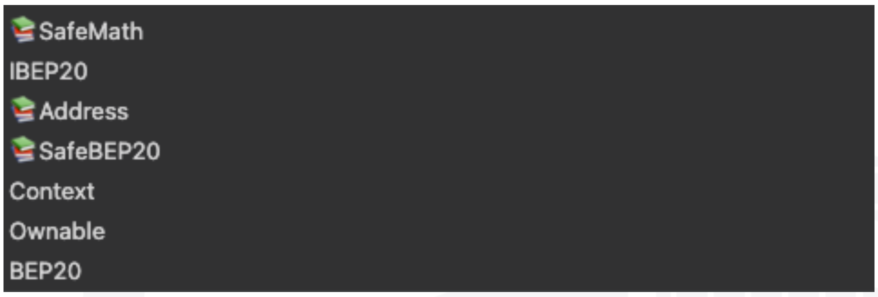
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

Imported packages:

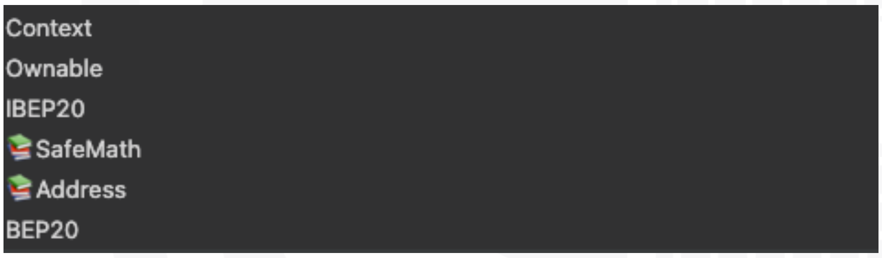
V1.0

MasterChef:



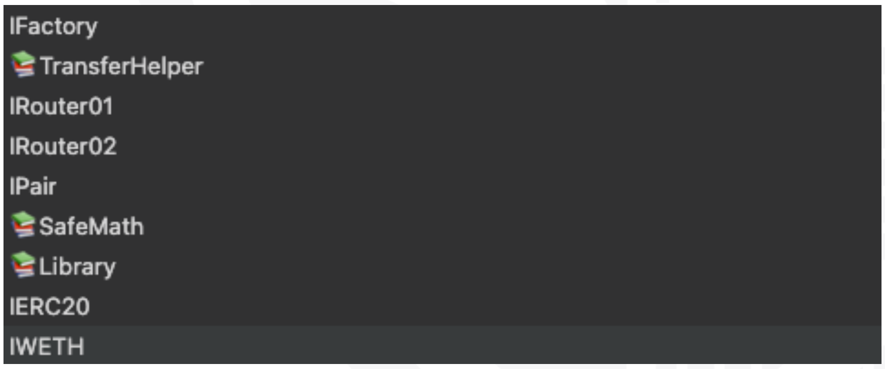
SafeMath
IBEP20
Address
SafeBEP20
Context
Ownable
BEP20

TokenAddress:



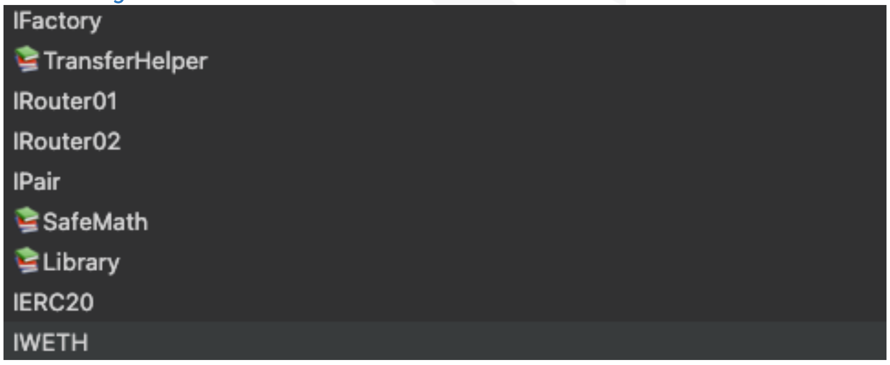
Context
Ownable
IBEP20
SafeMath
Address
BEP20

RouterAddress:



IFactory
TransferHelper
IRouter01
IRouter02
IPair
SafeMath
Library
IERC20
IWETH

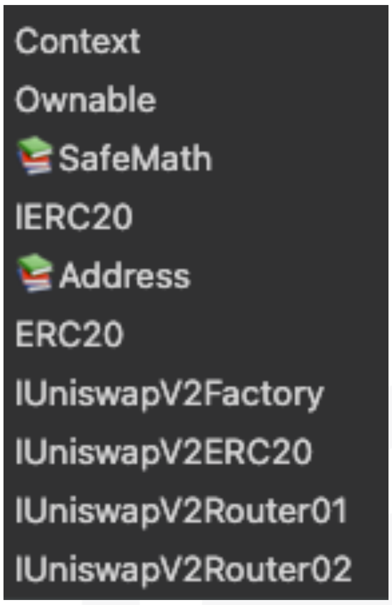
FactoryAddress:



IFactory
TransferHelper
IRouter01
IRouter02
IPair
SafeMath
Library
IERC20
IWETH

v2.0

TokenAddress

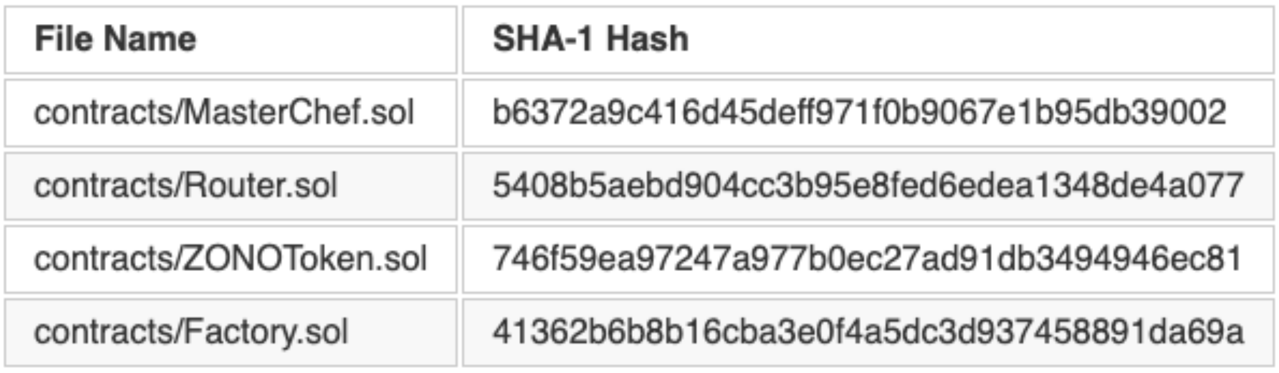

Context
Ownable
SafeMath
IERC20
Address
ERC20
IUniswapV2Factory
IUniswapV2ERC20
IUniswapV2Router01
IUniswapV2Router02

Tested Contract Files

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

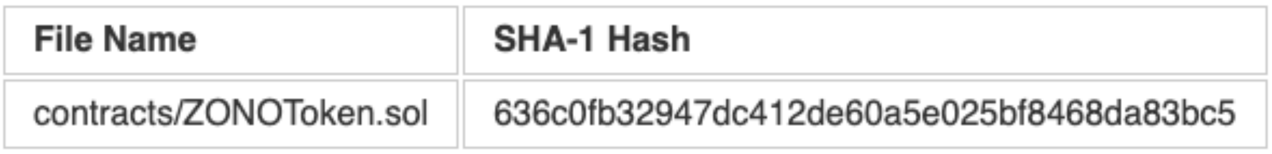
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0



File Name	SHA-1 Hash
contracts/MasterChef.sol	b6372a9c416d45deff971f0b9067e1b95db39002
contracts/Router.sol	5408b5aebd904cc3b95e8fed6edea1348de4a077
contracts/ZONOToken.sol	746f59ea97247a977b0ec27ad91db3494946ec81
contracts/Factory.sol	41362b6b8b16cba3e0f4a5dc3d937458891da69a

V2.0

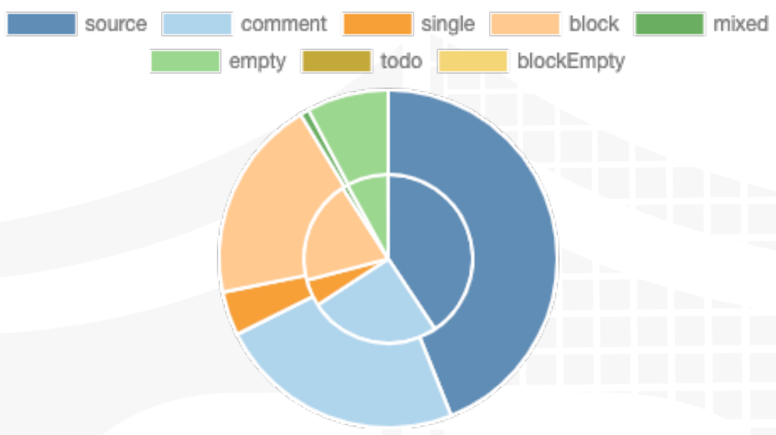


File Name	SHA-1 Hash
contracts/ZONOToken.sol	636c0fb32947dc412de60a5e025bf8468da83bc5

Metrics

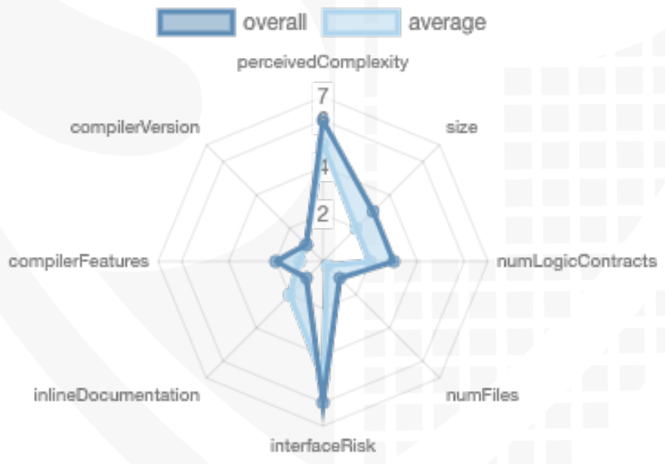
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	13	11	13	0

File (version)	Contracts	Libraries	Interfaces	Abstract
ZONOToken (2.0)	3	2	5	1

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	254	10

File (version)	Public	Payable
ZONOToken (2.0)	88	5

Version	External	Internal	Private	Pure	View
1.0	192	278	8	61	105

File (version)	External	Internal	Private	Pure	View
ZONOToken (2.0)	54	95	6	18	28

State Variables

Version	Total	Public
1.0	63	41

File (version)	Total	Public
ZONOToken (2.0)	33	21

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.6.12` `=0.6.6` `=0.5.16`		`yes`	`yes` (8 asm blocks)	

File (version)	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
ZONOToken (2.0)	`^0.6.0`		`yes`	`yes` (2 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	`yes`			`yes`	`yes`	`yes` `→ Assembly Call: Name: create2`

File (version)	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2
ZONOToken (2.0)	`yes`					

Scope of Work

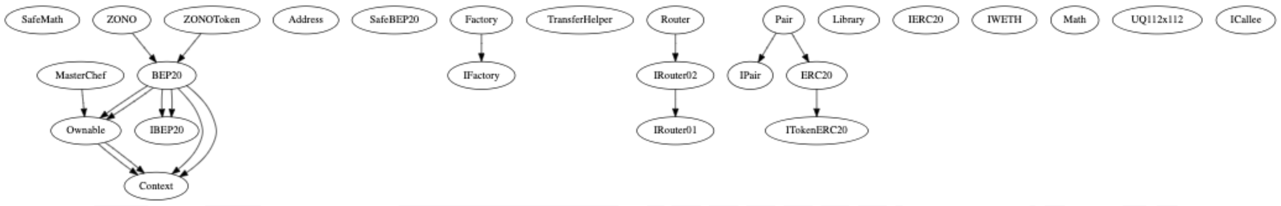
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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)

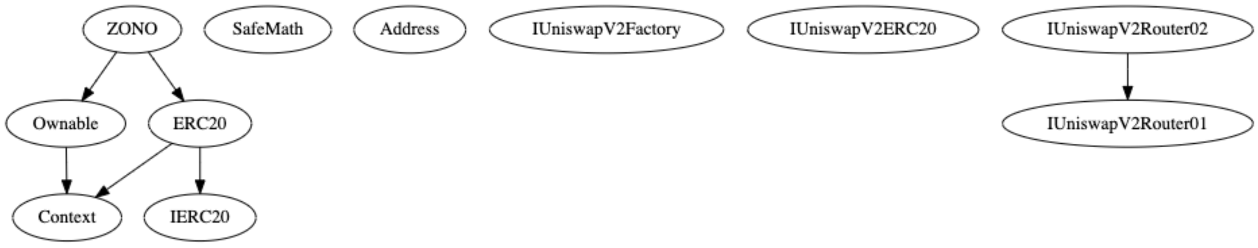
Inheritance Graph

v1.0



V2.0

ZONOToken



Verify Claims

Correct implementation of Token standard

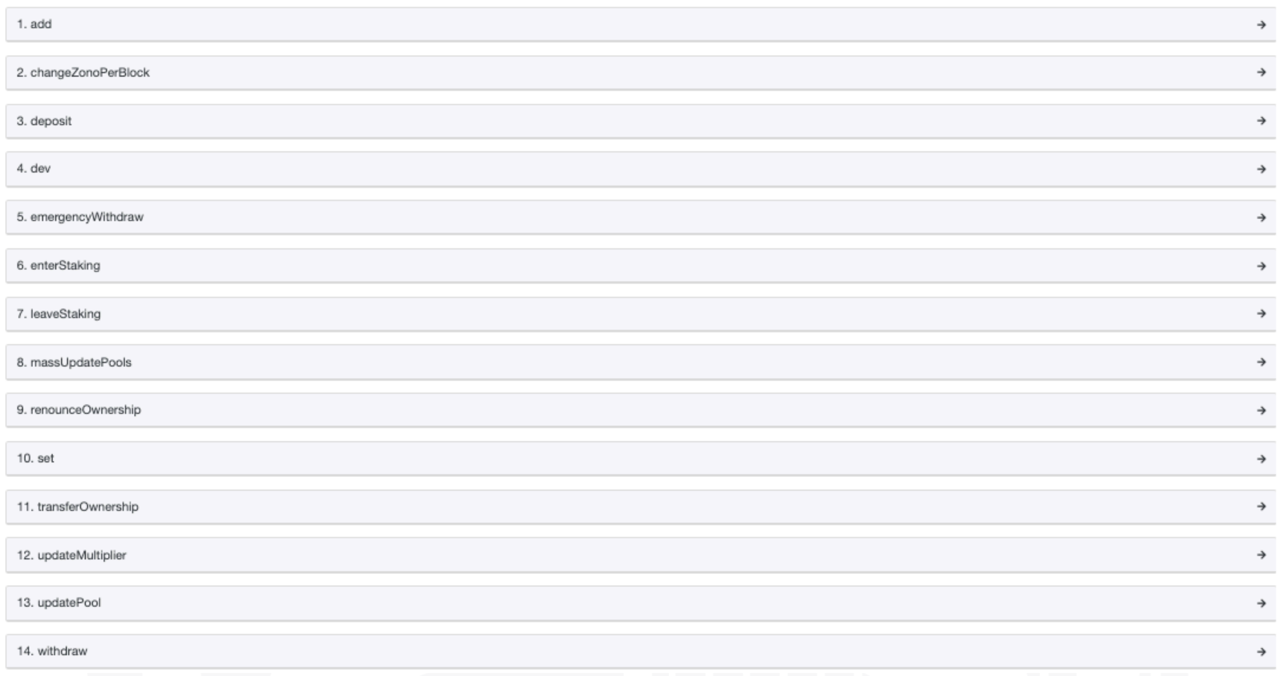
Tested	Verified
✓	✓

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	✓	✓	✓

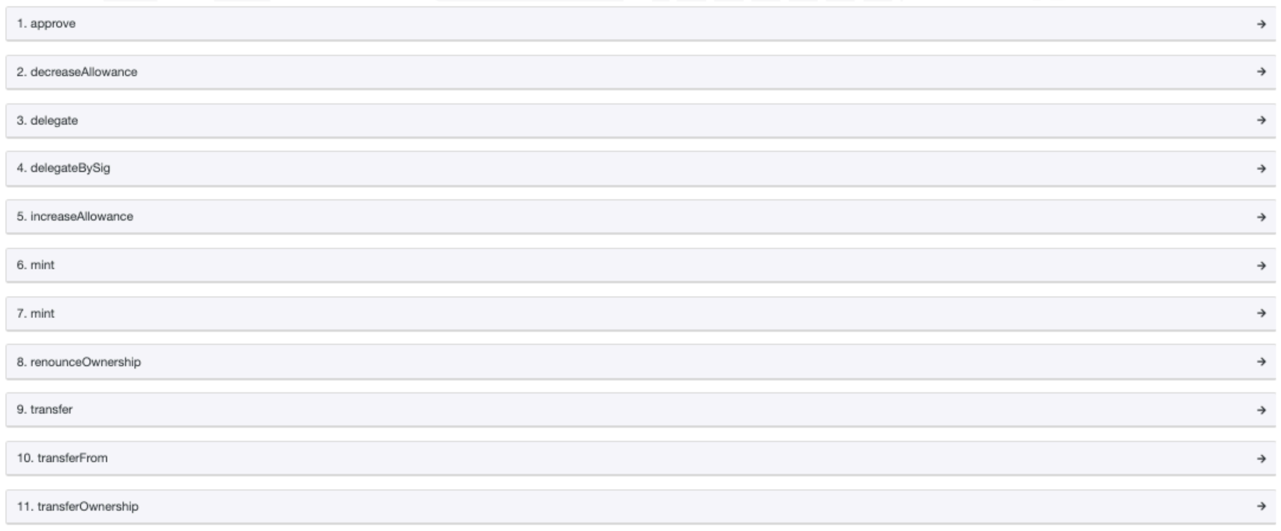
Write functions of contract

V1.0

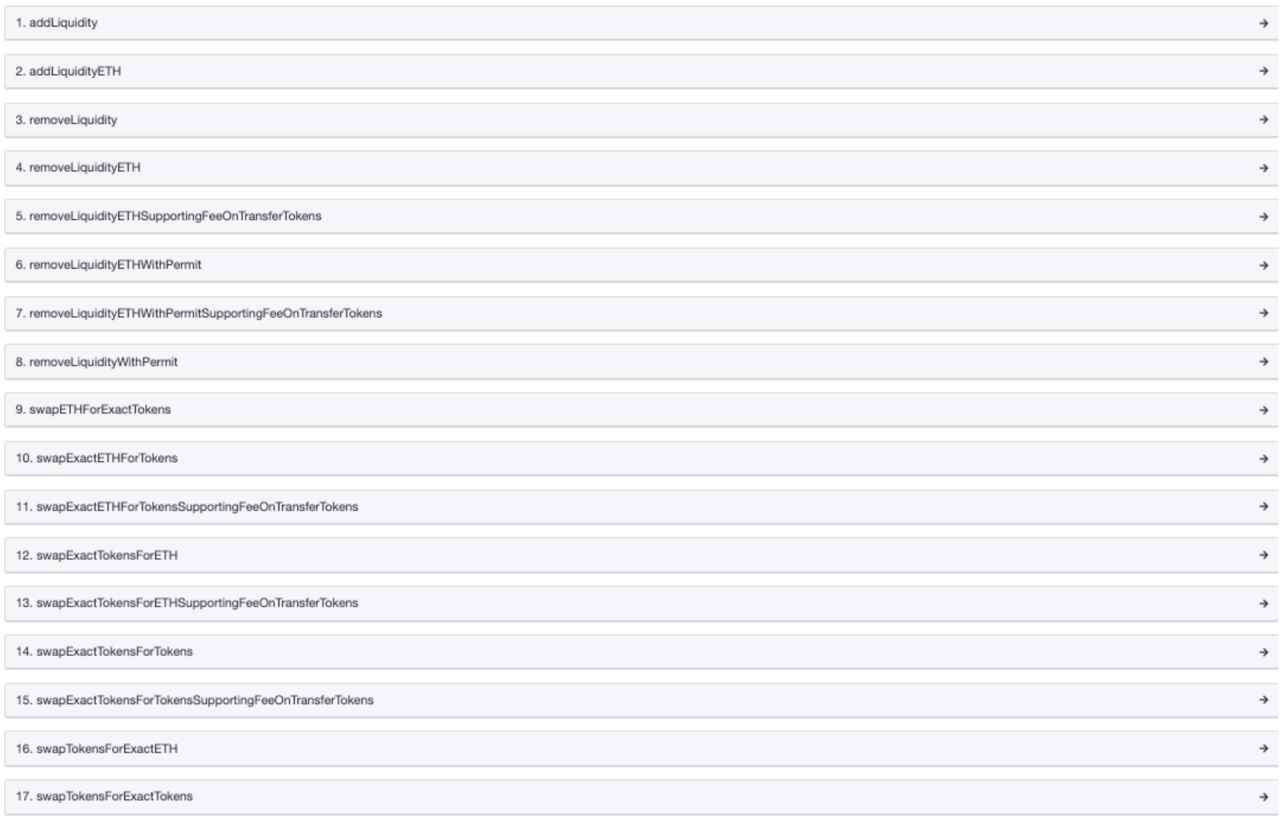
MasterChef:


1. add
2. changeZonoPerBlock
3. deposit
4. dev
5. emergencyWithdraw
6. enterStaking
7. leaveStaking
8. massUpdatePools
9. renounceOwnership
10. set
11. transferOwnership
12. updateMultiplier
13. updatePool
14. withdraw

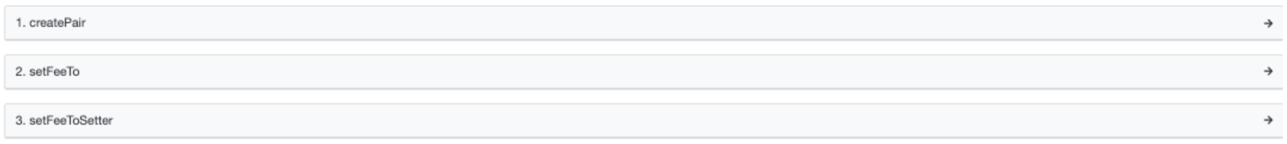
TokenAddress:


1. approve
2. decreaseAllowance
3. delegate
4. delegateBySig
5. increaseAllowance
6. mint
7. mint
8. renounceOwnership
9. transfer
10. transferFrom
11. transferOwnership

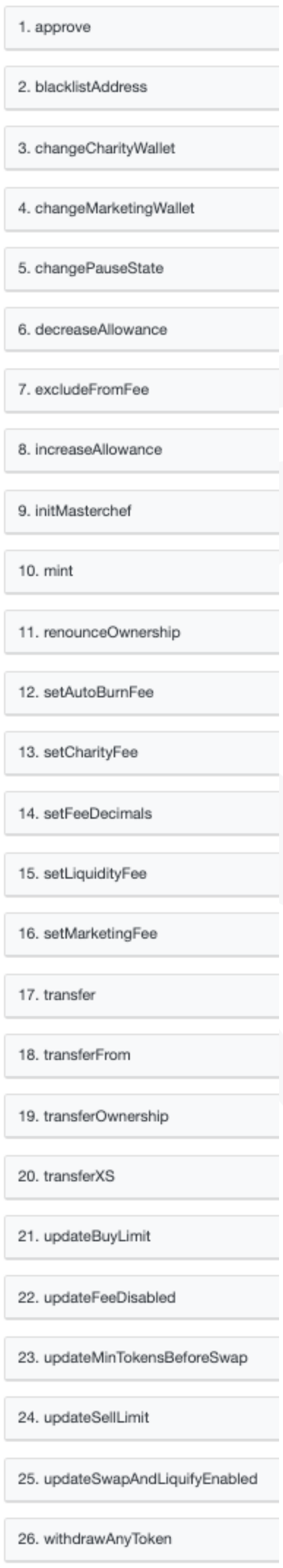
RouterAddress:


1. addLiquidity
2. addLiquidityETH
3. removeLiquidity
4. removeLiquidityETH
5. removeLiquidityETHSupportingFeeOnTransferTokens
6. removeLiquidityETHWithPermit
7. removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
8. removeLiquidityWithPermit
9. swapETHForExactTokens
10. swapExactETHForTokens
11. swapExactETHForTokensSupportingFeeOnTransferTokens
12. swapExactTokensForETH
13. swapExactTokensForETHSupportingFeeOnTransferTokens
14. swapExactTokensForTokens
15. swapExactTokensForTokensSupportingFeeOnTransferTokens
16. swapTokensForExactETH
17. swapTokensForExactTokens

FactoryAddress:


1. createPair
2. setFeeTo
3. setFeeToSetter

v2.0
ZONOToken



1. approve
2. blacklistAddress
3. changeCharityWallet
4. changeMarketingWallet
5. changePauseState
6. decreaseAllowance
7. excludeFromFee
8. increaseAllowance
9. initMasterchef
10. mint
11. renounceOwnership
12. setAutoBurnFee
13. setCharityFee
14. setFeeDecimals
15. setLiquidityFee
16. setMarketingFee
17. transfer
18. transferFrom
19. transferOwnership
20. transferXS
21. updateBuyLimit
22. updateFeeDisabled
23. updateMinTokensBeforeSwap
24. updateSellLimit
25. updateSwapAndLiquifyEnabled
26. withdrawAnyToken

Deployer cannot mint any new tokens

Version	File	Name	Exist	Tested	Verified
1.0	MasterChef	cannot mint	–	–	–
	TokenAddress	cannot mint	✓	✓	✗
	RouterAddress	cannot mint	–	–	–
	FactoryAddress	cannot mint	–	–	–
2.0	ZONOToken	cannot mint	✓	✓	✓

Max / Total Supply: 1.000.000.000

Comments:

v1.0

- Owner of **ZONOToken** (https://bscscan.com/address/0x8d08dcb48d59216ae5d0515aa6622c9beb42b76b#code) is **MasterChef** (https://bscscan.com/address/0x838648d92d314f743fe333d73bc9900e2f51d624) and not the provided source file **MasterChef** (https://bscscan.com/address/0xaff2b0031aa7a5c19c9cba9589817a704234d492#code)
    - Checked: Fri. 19. November 2021, 2:24 PM

V2.0

- ZONOToken
    - Only MasterChef can mint Tokens

Deployer cannot burn or lock user funds

Version	File	Name	Exist	Tested	Verified
1.0	MasterChef	Deployer cannot lock	✓	✓	✓
		Deployer cannot burn	–	–	–
	TokenAddress	Deployer cannot lock	✓	✓	✓
		Deployer cannot burn	✓	✓	✓
	RouterAddress	Deployer cannot lock	✓	✓	✓
		Deployer cannot burn	–	–	–
	FactoryAddress	Deployer cannot lock	✓	✓	✓
		Deployer cannot burn	–	–	–
2.0	ZONOToken	Deployer cannot lock	✓	✓	✗
		Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- RouterAddress
  - Uses IPair burn function
  - Uses IPair mint function
- MasterChef
  - Uses ZONO token to mint

V2.0

- If is from/to address is not excluded, fee is enabled and is not in swap and liquify
  - Transfer will burn tokens
- Deployer can lock user funds
  - If contract is paused by the owner
  - If address is blacklisted

Deployer cannot pause the contract

Version	File	Name	Exist	Tested	Verified
1.0	MasterChef	Deployer cannot pause	–	–	–
	TokenAddress	Deployer cannot pause	–	–	–
	RouterAddress	Deployer cannot pause	–	–	–
	FactoryAddress	Deployer cannot pause	–	–	–
2.0	ZONOToken	Deployer cannot pause	✓	✓	✗

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

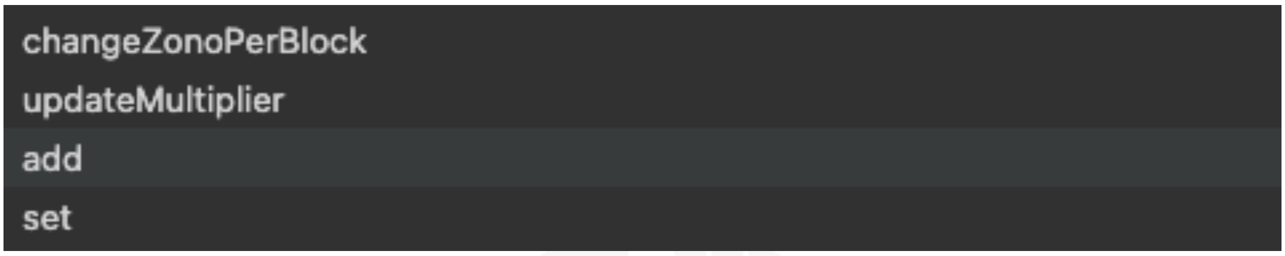
Legend

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

OnlyOwner functions

V1.0

MasterChef:

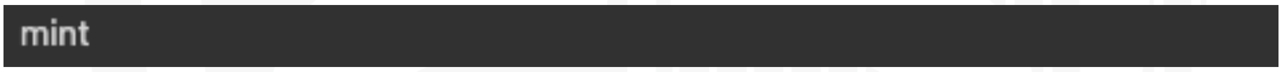


changeZonoPerBlock
updateMultiplier
add
set

Comments:

- changeZonoPerBlock
    - zonoPerBlock can be set without any limitations
- updateMulitplier
    - BONUS_MULTIPLIER can be set without any limitations

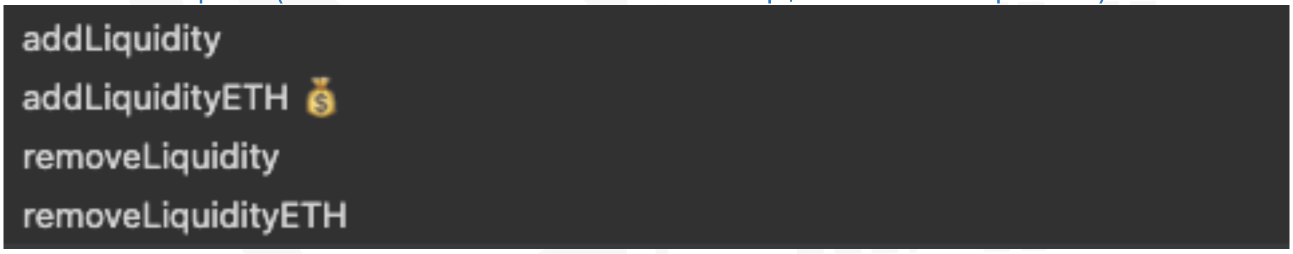
TokenAddress:



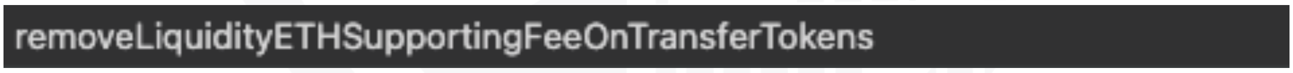
mint

RouterAddress:

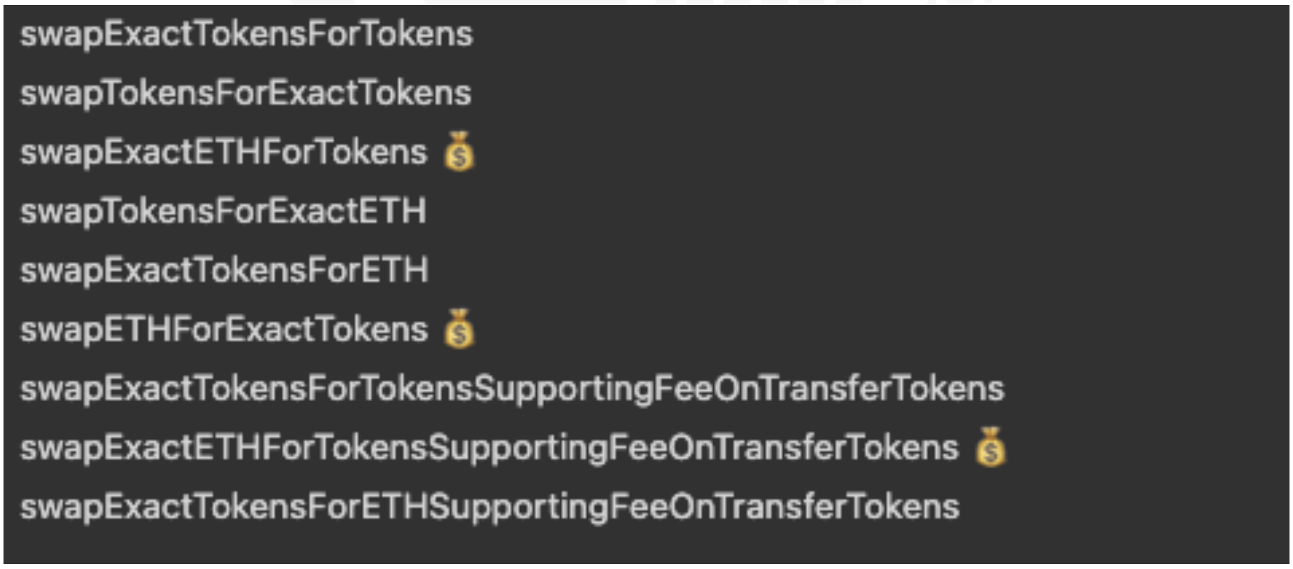
- Ensure modifier
    - require(deadline >= block.timestamp, “Router: Expired”)



addLiquidity
addLiquidityETH 💰
removeLiquidity
removeLiquidityETH



removeLiquidityETHSupportingFeeOnTransferTokens

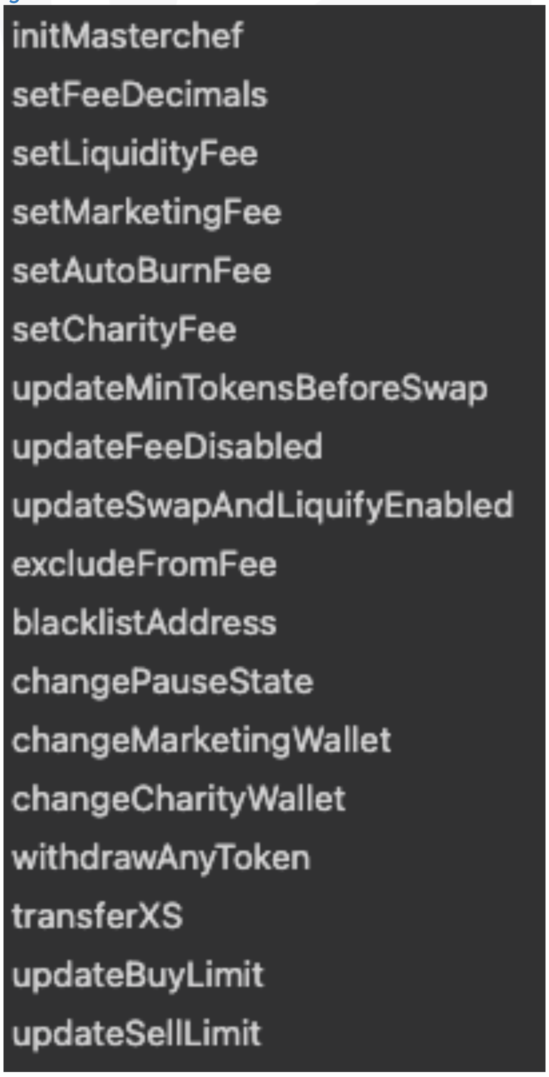


swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens 💰
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens 💰
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens 💰
swapExactTokensForETHSupportingFeeOnTransferTokens

Comments:
- addLiquidity
  - IPair is minting here
- addLiquidityETH (payable)
  - IPair is minting here
- removeLiquidity
  - IPair is burning here

FactoryAddress:
- Only FeeToSetter can set following state variables
  - feeTo
    - Is used in Pair contract
      - _mintFee function
  - feeToSetter

v2.0
- forMint
  - Mint
- onlyOwner



```
initMasterchef
setFeeDecimals
setLiquidityFee
setMarketingFee
setAutoBurnFee
setCharityFee
updateMinTokensBeforeSwap
updateFeeDisabled
updateSwapAndLiquifyEnabled
excludeFromFee
blacklistAddress
changePauseState
changeMarketingWallet
changeCharityWallet
withdrawAnyToken
transferXS
updateBuyLimit
updateSellLimit
```

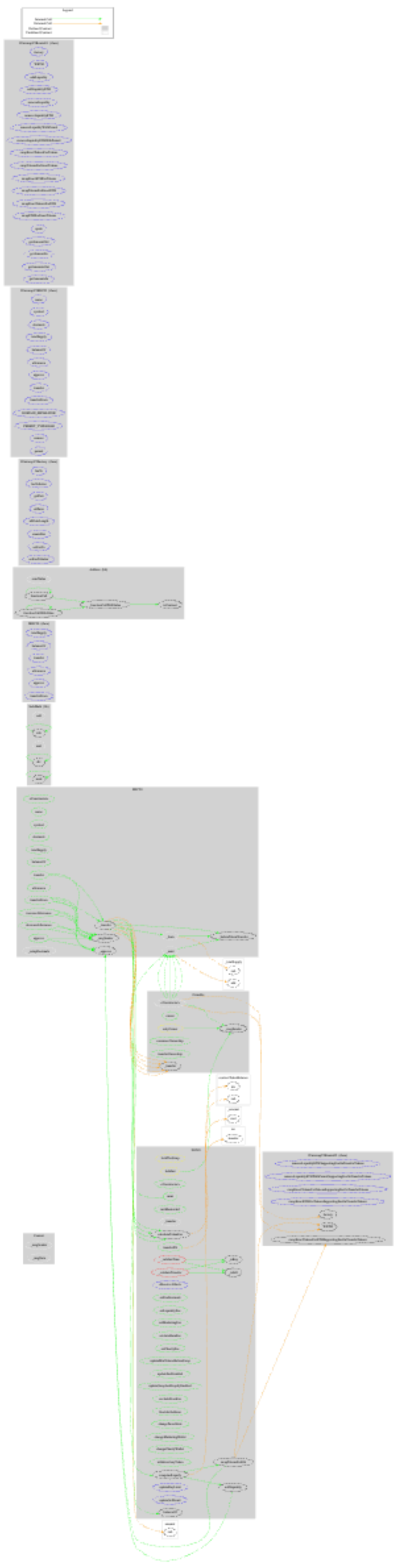
Comments:

- Following fees can be set without any restriction
    - liquidityFee
    - _marketingFee
    - _autoBurn
    - _charity
- Owner can pause contract
- Owner can blacklist addresses
- transferXS
    - Owner can transfer address balance to a certain address
- buyLimit and sellLimit can be set without any limitations

CallGraph

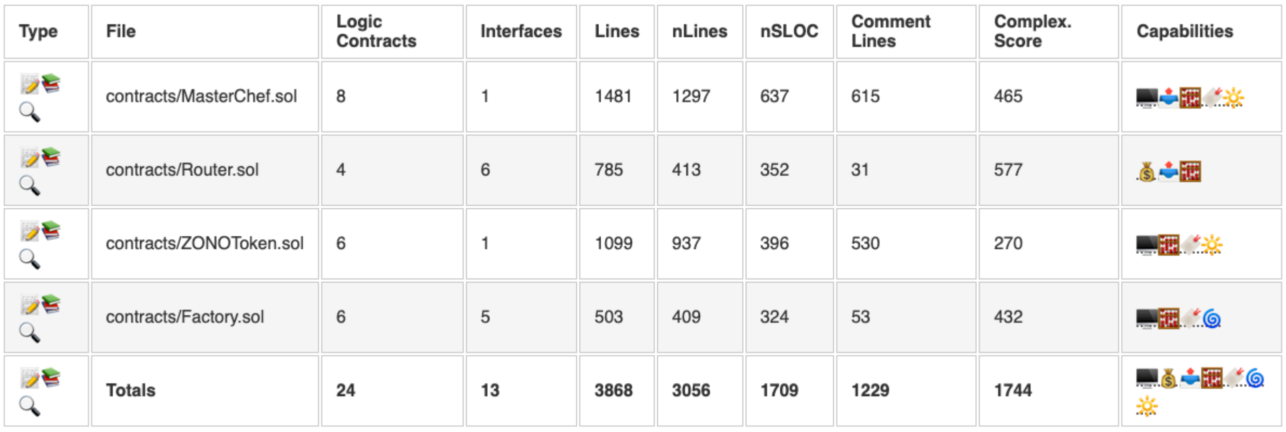
v1.0

v2.0



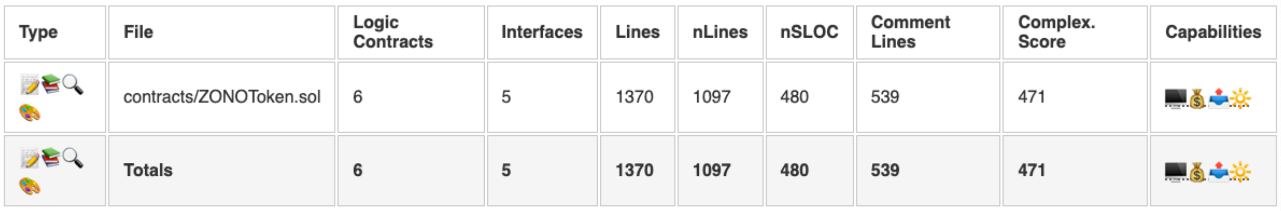
Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍	contracts/MasterChef.sol	8	1	1481	1297	637	615	465	🖥📥🧮☄️☀️
📝📚🔍	contracts/Router.sol	4	6	785	413	352	31	577	💰📥🧮
📝📚🔍	contracts/ZONOToken.sol	6	1	1099	937	396	530	270	🖥🧮☄️☀️
📝📚🔍	contracts/Factory.sol	6	5	503	409	324	53	432	🖥🧮☄️🌀
📝📚🔍	**Totals**	**24**	**13**	**3868**	**3056**	**1709**	**1229**	**1744**	🖥💰📥🧮☄️🌀☀️

V2.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/ZONOToken.sol	6	5	1370	1097	480	539	471	🖥💰📥☀️
📝📚🔍🎨	**Totals**	**6**	**5**	**1370**	**1097**	**480**	**539**	**471**	🖥💰📥☀️

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	Factory Address	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	MasterC hef	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
#3	RouterA ddress	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
#4	TokenA ddress	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

#5	Factory Address	Missing Zero Address Validation (missing-zero-check)	469, 494, 499, 319	Check that the address is not zero
#6	MasterChef	Missing Zero Address Validation (missing-zero-check)	1257, 1477	Check that the address is not zero
#7	RouterAddress	Missing Zero Address Validation (missing-zero-check)	362	Check that the address is not zero
#8	ZONOToken (V2.0)	Missing Zero Address Validation (missing-zero-check)	1347, 1343, 1079, 1357	Check that the address is not zero
#9	ZONOToken (V2.0)	Unused return values	1242	Ensure that all the return values of the function calls are used
#10	ZONOToken (V2.0)	State variable visibility is not set.	989, 1009, 1020, 1021,	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	RouterAddress	Functions that are not used	26	Remove unused functions
#2	ZONOToken (V2.0)	State variables that could be declared constant	1003, 1002, 1018	Add the `constant` attributes to state variables that never change
#3	ZONOToken (V2.0)	Functions that are not used	1189, 1194, 1172, 1156	Remove unused functions
#4	ZONOToken (V2.0)	Unused state variables	534, 1003	Remove unused state variables

Commented Code exist

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

File	Line	Comment
MasterChef	136	// assert(a == b * c + a % b); // There is no case in which this doesn't hold

RouterAddress	27, 33	// bytes4(keccak256(bytes('approve(address,uint256)')));
RouterAddress	39	// bytes4(keccak256(bytes('transferFrom(address,address,uint256)')));
ZONOToken (V2.0)	225	// assert(a == b * c + a % b); // There is no case in which this doesn't hold
ZONOToken (V2.0)	1064	//_marketingWallet = msg.sender;
ZONOToken (V2.0)	1068	//_charityWallet = msg.sender;

Recommendation

Remove the commented code, or address them properly.

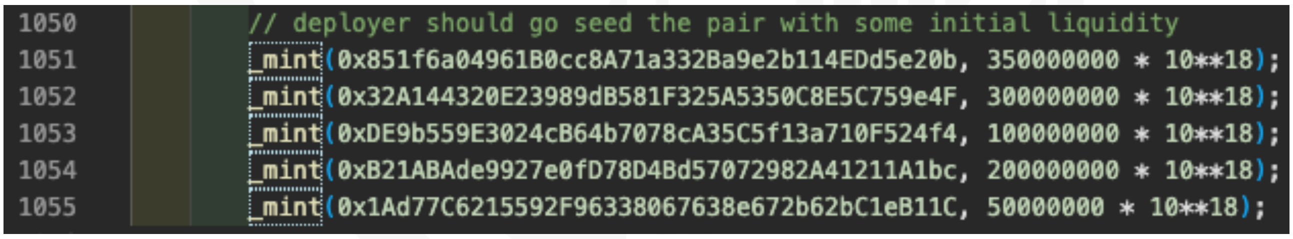
Audit Comments

19. November 2021:

For more information read report

04. December 2021

- IBEP interface changed to IERC20
- BEP20 changed to ERC20
- Uniswap added
- Minting for 5 addresses in constructor
  - 1.000.000.000



```
// deployer should go seed the pair with some initial liquidity
_mint(0x851f6a04961B0cc8A71a332Ba9e2b114EDd5e20b, 350000000 * 10**18);
_mint(0x32A144320E23989dB581F325A5350C8E5C759e4F, 300000000 * 10**18);
_mint(0xDE9b559E3024cB64b7078cA35C5f13a710F524f4, 100000000 * 10**18);
_mint(0xB21ABAde9927e0fD78D4Bd57072982A41211A1bc, 200000000 * 10**18);
_mint(0x1Ad77C6215592F96338067638e672b62bC1eB11C, 50000000 * 10**18);
```

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	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	NOT 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	**NOT PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**NOT 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**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY