



**SOLIDProof**  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

**ZonoSwap**

**Audit**

**Security Assessment**

**27. April, 2022**

**For**



**ZONOSWAP**

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	25
Critical issues	26
High issues	26
Medium issues	26
Low issues	26
Informational issues	27
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 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	27. April 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Binance Smart Chain (BEP20)

## **Website**

<https://zonoswap.com/#/swap>

## **Telegram**

<https://t.me/zonoswap>

## **Twitter**

<https://twitter.com/ZonoSwap>

## **Facebook**

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

## **Github**

<https://github.com/zono-swap>

## 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 22nd of April 2022, **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



## Contract Link

### v1.0

- ZonoV3Token
  - <https://bscscan.com/address/0xa8de787e6196a991a35e46a5b8d5505e15b486ad#code>
- ZonoV3MasterChef
  - <https://bscscan.com/address/0xec1b4a45dce66b0ef8ea95e99ce152312378e166>
- ZonoV3Lottery
  - <https://bscscan.com/address/0x823ca204d56a30a95fcfca6682d4ac54ad9fea62>

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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:

Lottery

```
Context
Ownable
ReentrancyGuard
SafeMath
IERC20
Address
SafeERC20
IRandomNumberGenerator
IZonoV3Lottery
```

Token

```
Context
Ownable
SafeMath
IERC20
AntiBotHelper
FeeHelper
Address
ERC20
MintableERC20
IUniswapV2Factory
IUniswapV2Router01
IUniswapV2Router02
```

MasterChef

```
Address
SafeERC20
Context
Ownable
ReentrancyGuard
SafeMath
IMintableERC20
```



## Tested Contract Files

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

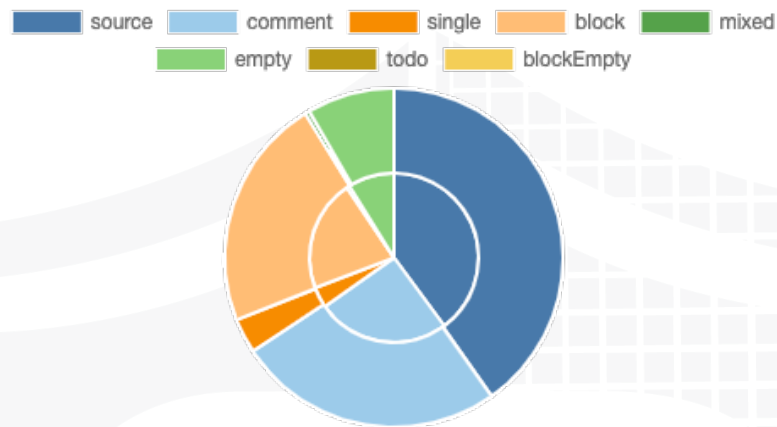
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

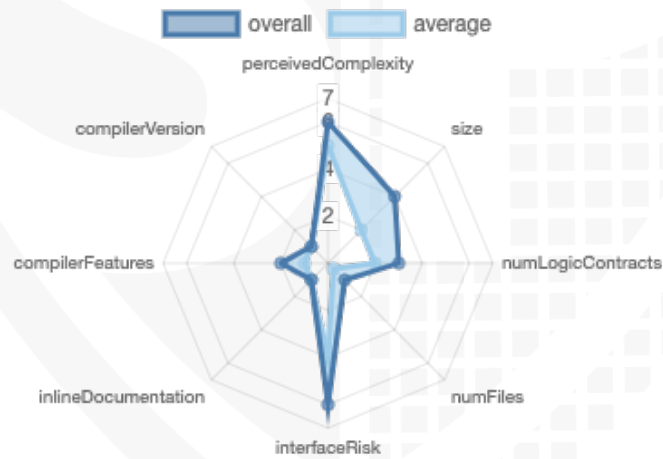
File Name	SHA-1 Hash
contracts/ZonoV3Lottery.sol	01a58cc3d1e69a3eaccec0078c55d7562efcce57
contracts/ZonoV3Token.sol	0ad357cb40f95c1616d985be8de332e38744b961
contracts/ZonoV3MasterChef.sol	1aabcd3daabcabee966ee4b3c91218ff3a46bad4

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	7	8	9	8

### 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	151	6

Version	External	Internal	Private	Pure	View
1.0	123	237	12	42	66

## State Variables

Version	Total	Public
1.0	77	44

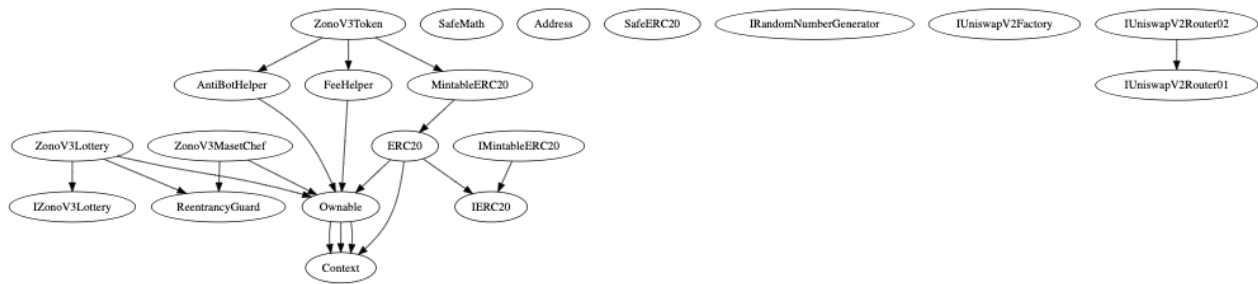
## Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.8.0  &gt;=0.4.0  ^0.8.4  &gt;=0.5.0  &gt;=0.6.2</code>		yes	yes (7 asm blocks)	

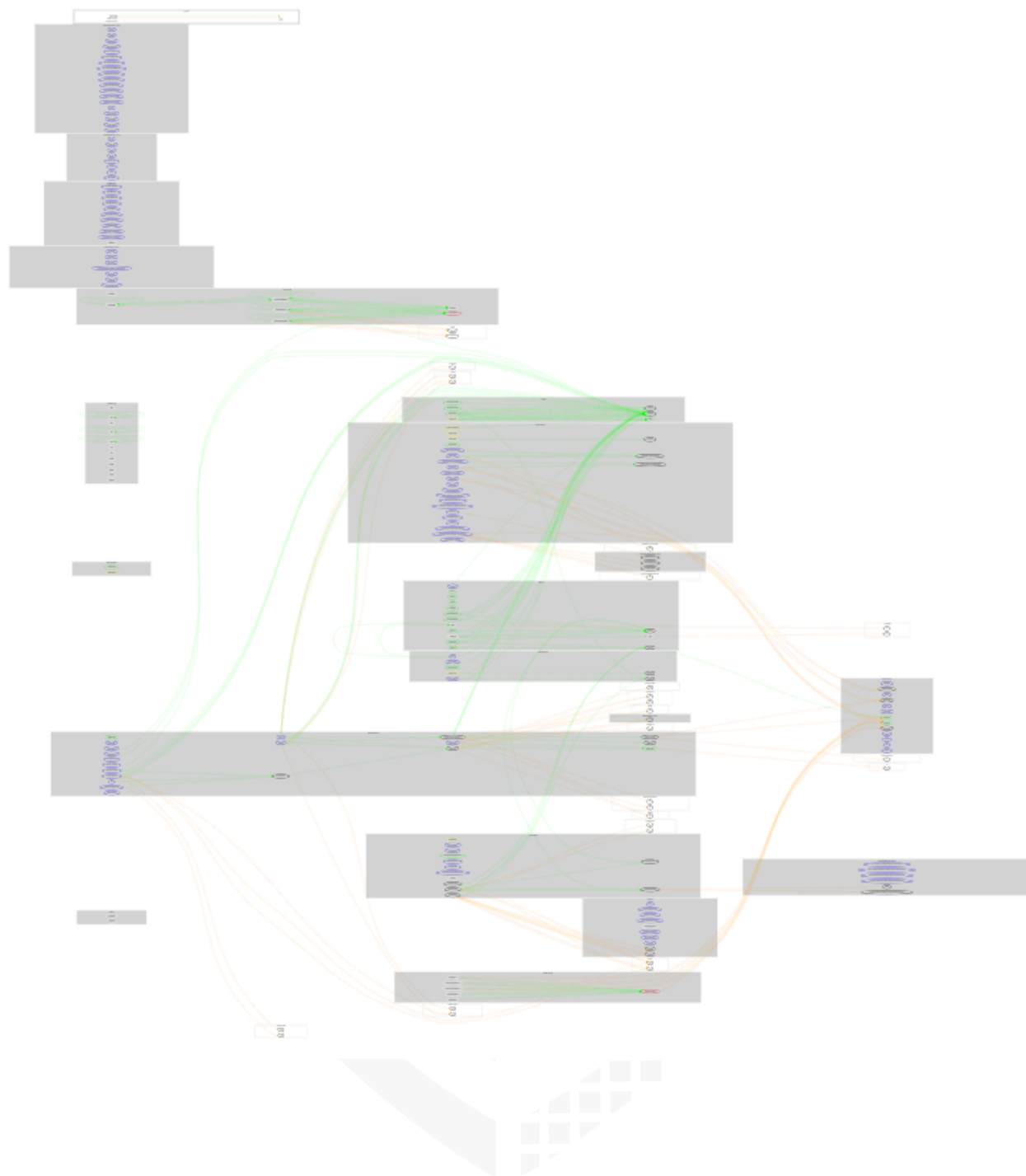
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes		yes	yes		

# 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).

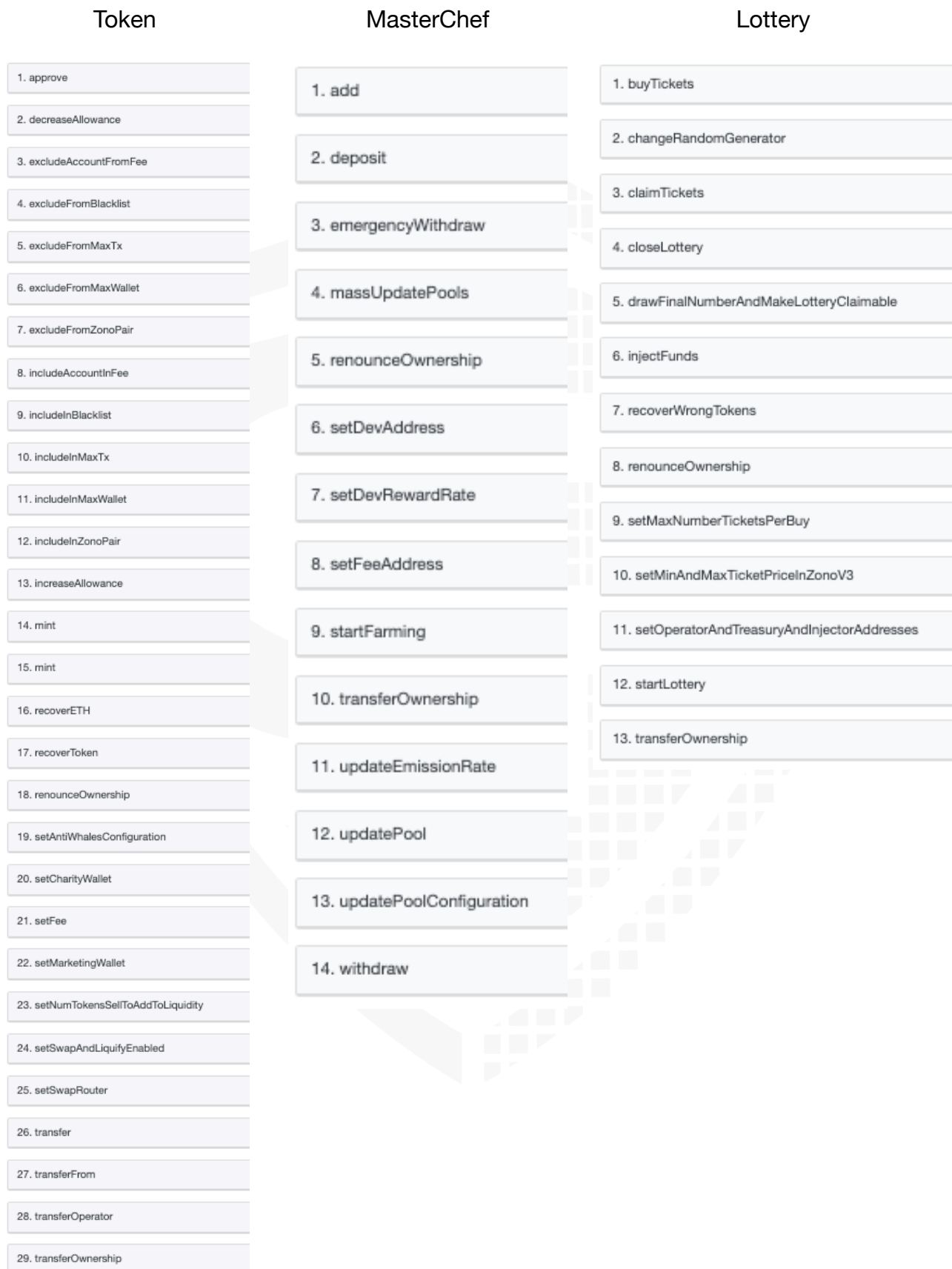
We will verify the following claims:

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





## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

### v1.0

- Tokens will be minted while updating pool
- OnlyOperator can mint new token to
  - anyone
  - Himself

## Deployer cannot burn or lock user funds

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

Comments:

### v1.0

- Token will be burned while tx
- Owner can lock user funds by setting address to blacklist

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



## 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

### Token

- ▼  setSwapRouter
    - Ⓜ onlyOwner
  - ▼  setSwapAndLiquifyEnabled
    - Ⓜ onlyOwner
  - ▼  setMarketingWallet
    - Ⓜ onlyOwner
  - ▼  setCharityWallet
    - Ⓜ onlyOwner
  - ▼  setNumTokensSellToAddToLiquidity
    - Ⓜ onlyOwner
- ▼  excludeFromBlacklist
    - Ⓜ onlyOwner
  - ▼  includeInBlacklist
    - Ⓜ onlyOwner
  - ▼  excludeFromMaxTx
    - Ⓜ onlyOwner
  - ▼  includeInMaxTx
    - Ⓜ onlyOwner
  - ▼  excludeFromMaxWallet
    - Ⓜ onlyOwner
  - ▼  includeInMaxWallet
    - Ⓜ onlyOwner
  - ▼  setAntiWhalesConfiguration
    - Ⓜ onlyOwner
- ▼  mint
    - Ⓜ onlyOperator
  - ▼  transferOperator
    - Ⓜ onlyOperator
  - ▼  recoverToken
    - Ⓜ onlyOwner
  - ▼  recoverETH
    - Ⓜ onlyOwner

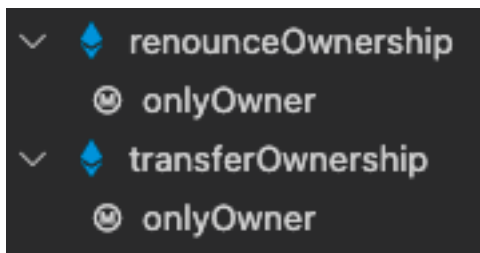
### MasterChef

- ▼  startFarming
  - Ⓜ onlyOwner
- ▼  add
  - Ⓜ onlyOwner
- ▼  updatePoolConfiguration
  - Ⓜ onlyOwner
-  massUpdatePools
-  updatePool
- ▼  deposit
  - Ⓜ nonReentrant
- ▼  withdraw
  - Ⓜ nonReentrant
- ▼  emergencyWithdraw
  - Ⓜ nonReentrant
- ▼  setFeeAddress
  - Ⓜ onlyOwner
- ▼  setDevAddress
  - Ⓜ onlyOwner
- ▼  setDevRewardRate
  - Ⓜ onlyOwner
- ▼  updateEmissionRate
  - Ⓜ onlyOwner

- ▼ 🔹 setFee
  - Ⓜ onlyOwner
- ▼ 🔹 excludeAccountFromFee
  - Ⓜ onlyOwner
- ▼ 🔹 includeAccountInFee
  - Ⓜ onlyOwner
- ▼ 🔹 excludeFromZonoPair
  - Ⓜ onlyOwner
- ▼ 🔹 includeInZonoPair
  - Ⓜ onlyOwner

## Lottery

- ▼ 🔹 buyTickets
  - Ⓜ notContract
  - Ⓜ nonReentrant
- ▼ 🔹 claimTickets
  - Ⓜ notContract
  - Ⓜ nonReentrant
- ▼ 🔹 closeLottery
  - Ⓜ onlyOperator
  - Ⓜ nonReentrant
- ▼ 🔹 drawFinalNumberAndMakeLotteryClaimable
  - Ⓜ onlyOperator
  - Ⓜ nonReentrant
- ▼ 🔹 changeRandomGenerator
  - Ⓜ onlyOwner
- ▼ 🔹 injectFunds
  - Ⓜ onlyOwnerOrInjector
- ▼ 🔹 startLottery
  - Ⓜ onlyOperator
- ▼ 🔹 recoverWrongTokens
  - Ⓜ onlyOwner
- ▼ 🔹 setMinAndMaxTicketPriceInZonoV3
  - Ⓜ onlyOwner
- ▼ 🔹 setMaxNumberTicketsPerBuy
  - Ⓜ onlyOwner
- ▼ 🔹 setOperatorAndTreasuryAndInjectorAddresses
  - Ⓜ onlyOwner



## Comments

- Deployer can set following state variables without any limitations
  - Token
    - \_numTokensSellToAddToLiquidity
  - Lottery
    - maxNumberTicketsPerBuyOrClaim
    - minPriceTicketInZonoV3
    - maxPriceTicketInZonoV3
  - MasterChef
    - zonoPerBlock
- Deployer can enable/disable following state variables
  - Token
    - \_swapAndLiquifyEnabled
    - \_isExcludedFromMaxWallet
    - \_isExcludedFromMaxTx
    - \_blacklist
    - \_blacklist
    - \_isExcludedFromFee
    - \_isZonoPair
- Deployer can set following addresses
  - Token
    - \_swapRouter
    - \_marketingWallet
    - \_charityWallet
    - \_operator
  - Lottery
    - randomGenerator
    - operatorAddress
    - treasuryAddress
    - injectorAddress
    - \_owner
    -
  - MasterChef

- feeAddress
- devAddress
- We recommend you to use an external library like VRF to generate a random number. For more information please read the following: <https://docs.chain.link/docs/chainlink-vrf/>
- In the lottery contract is more than one authority to check or call some functions
  - Owner
  - injectorAddress
  - Operator
- If owner is renounced there are actually an authority who can inject funds for example
  - In the contracts there are several powers to do something, although the owner has renounced his property. However, the injector can call some functions that are basically not callable after the renunciation.
- Wrong comment. MinPrice must be lower than maxPrice but the require statement is wrong. You are checking for “equal lower” instead of lower.
  - If the minPrice and maxPrice is set to 0, the amountTonoV3ToTransfer will be 0

**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/ZonoV3Lottery.sol	7	3	1845	1508	847	617	452	
	contracts/ZonoV3Token.sol	9	4	1738	1409	714	625	539	
	contracts/ZonoV3MasterChef.sol	7	2	1456	1231	688	504	379	
	<b>Totals</b>	<b>23</b>	<b>9</b>	<b>5039</b>	<b>4148</b>	<b>2249</b>	<b>1746</b>	<b>1370</b>	

## 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

No low issues

Issue	File	Type	Line	Description
#1	Main	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	All	A floating pragma is set	At the top of source code	The current pragma Solidity directive is „^0.“.
#3	Token	State variable visibility is not set	1474	It is best practice to set the visibility of state variables explicitly
#4	Token	State variables shadowing	1645	Rename the state variables that shadow another component
#5	MasterC hef	Missing Events Arithmetic	1059, 1107, 1536	Emit an event for critical parameter changes

## Informational issues

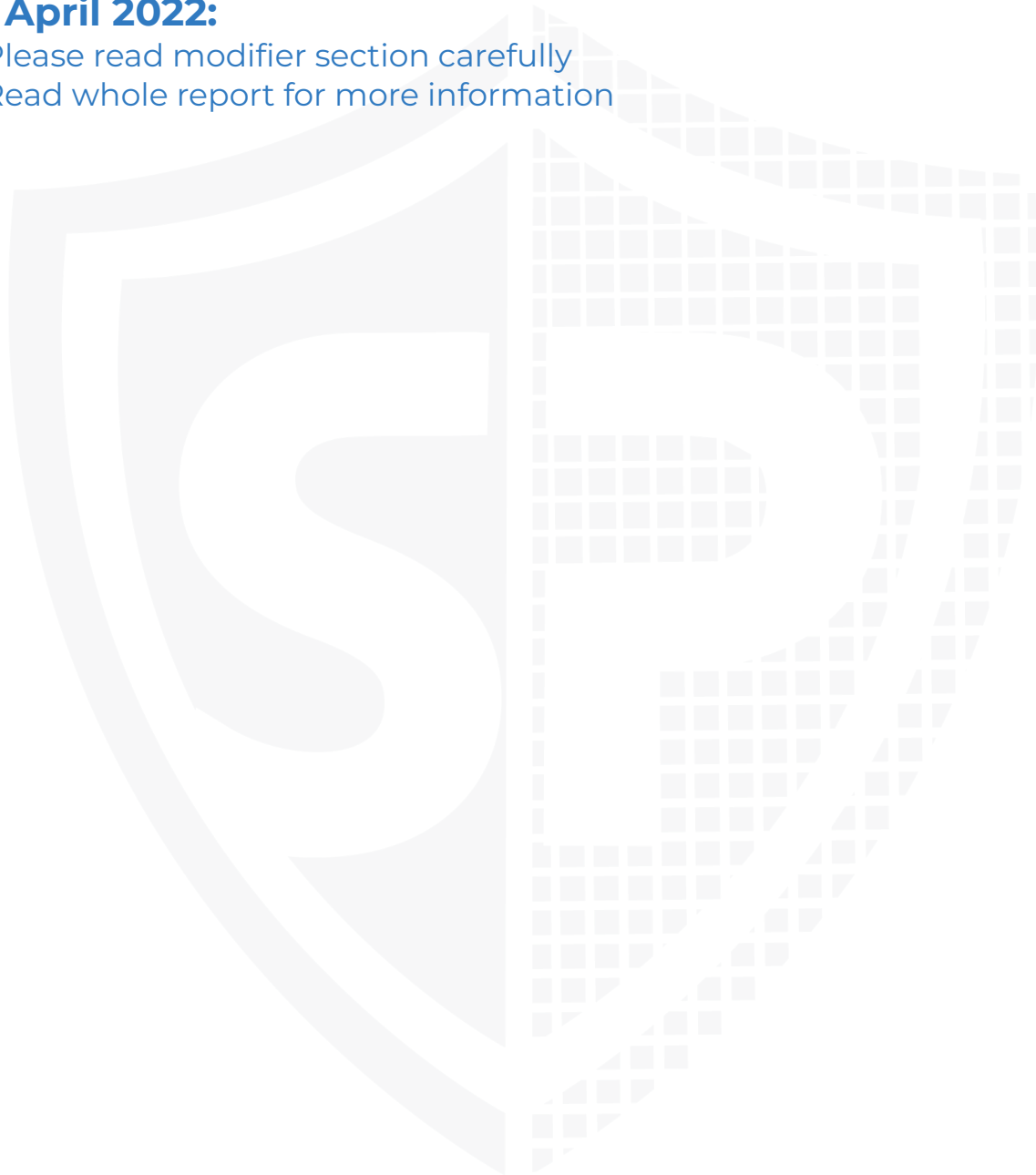
Issue	File	Type	Line	Description
#1	MasterC hef	State variables that could be declared constant (constable-states)	900	Add the `constant` attributes to state variables that never change
#2	MasterC hef	Unused state variables	900	Remove unused state variables
#3	MasterC hef	Misspelling	See description	Change following words:  - vairables L1179  Make sure to change it everywhere else as well.
#4	Token	Misspelling	See description	Change following words:  - MarketingFeeTrasferred L1481 - recieve L1503 - swaping L1503  Make sure to change it everywhere else as well.
#5	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.

## 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.

### 27. April 2022:

- Please read modifier section carefully
- Read whole report for more information



## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

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

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

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



The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem with a grid-like pattern, rendered in a darker shade of blue. The entire composition is set against a solid blue background.

Solid  
Proofed

**Blockchain Security | Smart Contract Audits | KYC**

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

MADE IN GERMANY