# ZonoSwap

# Audit

## Security Assessment
## 30. April, 2022

For

ZONOSWAP

# Disclaimer

| Version | Date | Description |
|---------|------|-------------|
| 1.0 | 27. April 2022 | • Layout project<br>• Automated- /Manual-Security Testing<br>• Summary |

**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 itselt.

# 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

- ZonoV3Factory
    - https://bscscan.com/address/0xaa3ab8e218e364b477e464aa494b498b41565ad3#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:

### 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
```

### Factory

```
Context
Ownable
IBEP20
SafeMath
ReentrancyGuard
Address
SafeBEP20
ZonoV3Pool
```

# 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/ZonoV3PoolFactory.sol | 0ff347ed324b932dbcfc9555f138257b5c6cb188 |
| contracts/ZonoV3Lottery.sol | 01a58cc3d1e69a3eaccec0078c55d7562efcce57 |
| contracts/ZonoV3Token.sol | 0ad357cb40f95c1616d985be8de332e38744b961 |
| contracts/ZonoV3MasterChef.sol | 1aabcd3daabcabee966ee4b3c91218ff3a46bad4 |

# Metrics

## Source Lines
### v1.0
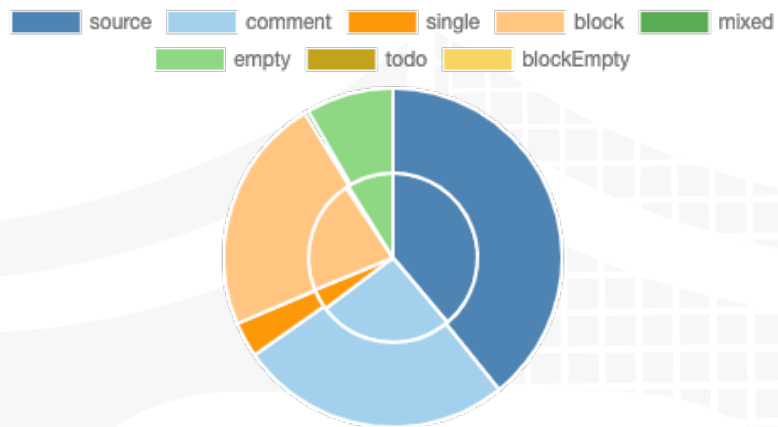


## Risk Level
### v1.0
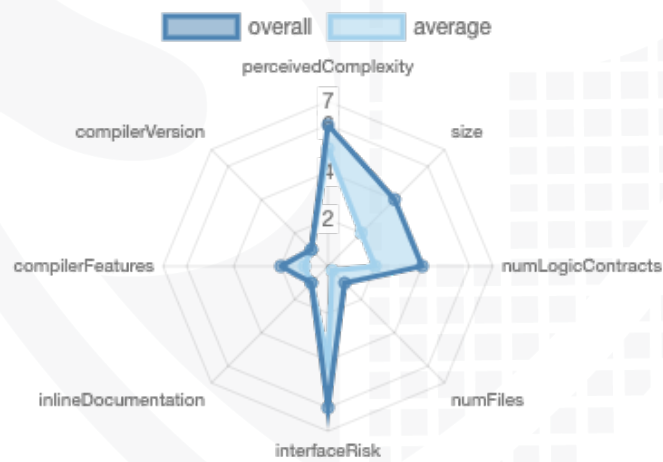
# Capabilities

## Components

| Version | Contracts | Libraries | Interfaces | Abstract |
|---------|-----------|-----------|------------|----------|
| 1.0 | 9 | 11 | 10 | 11 |

## 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 | 180 | 6 |

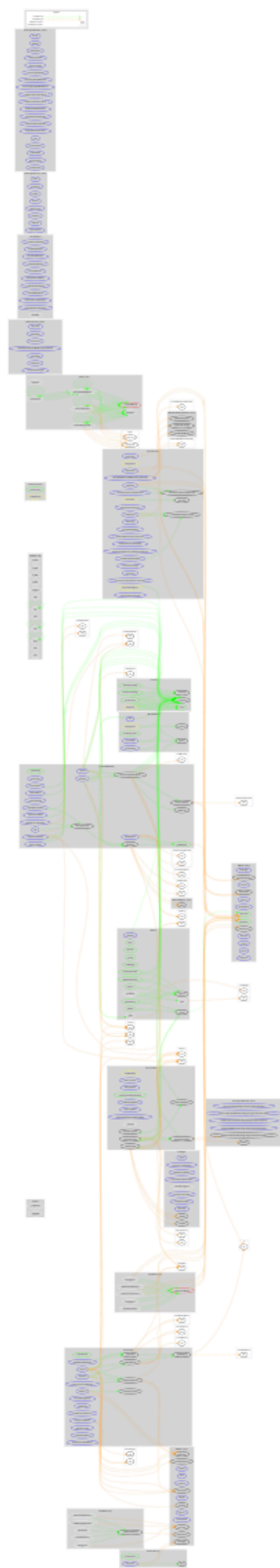| Version | External | Internal | Private | Pure | View |
|---|---|---|---|---|---|
| 1.0 | 146 | 309 | 14 | 56 | 81 |

## State Variables

| Version | Total | Public |
|---|---|---|
| 1.0 | 98 | 61 |

## Capabilities

| Version | Solidity Versions observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---|---|---|---|---|---|
| 1.0 | >=0.6.0 <0.8.0 >=0.4.0 >=0.6.2 <0.8.0 ^0.6.0 0.6.12 ^0.8.0 ^0.8.4 >=0.5.0 >=0.6.2 | | yes | yes (10 asm blocks) | |

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

| 1.0 | yes | | yes | yes | | yes →Asse mblyCa ll:Nam e:crea te2 |
|------|-----|-----|-----|-----|-----|-----|

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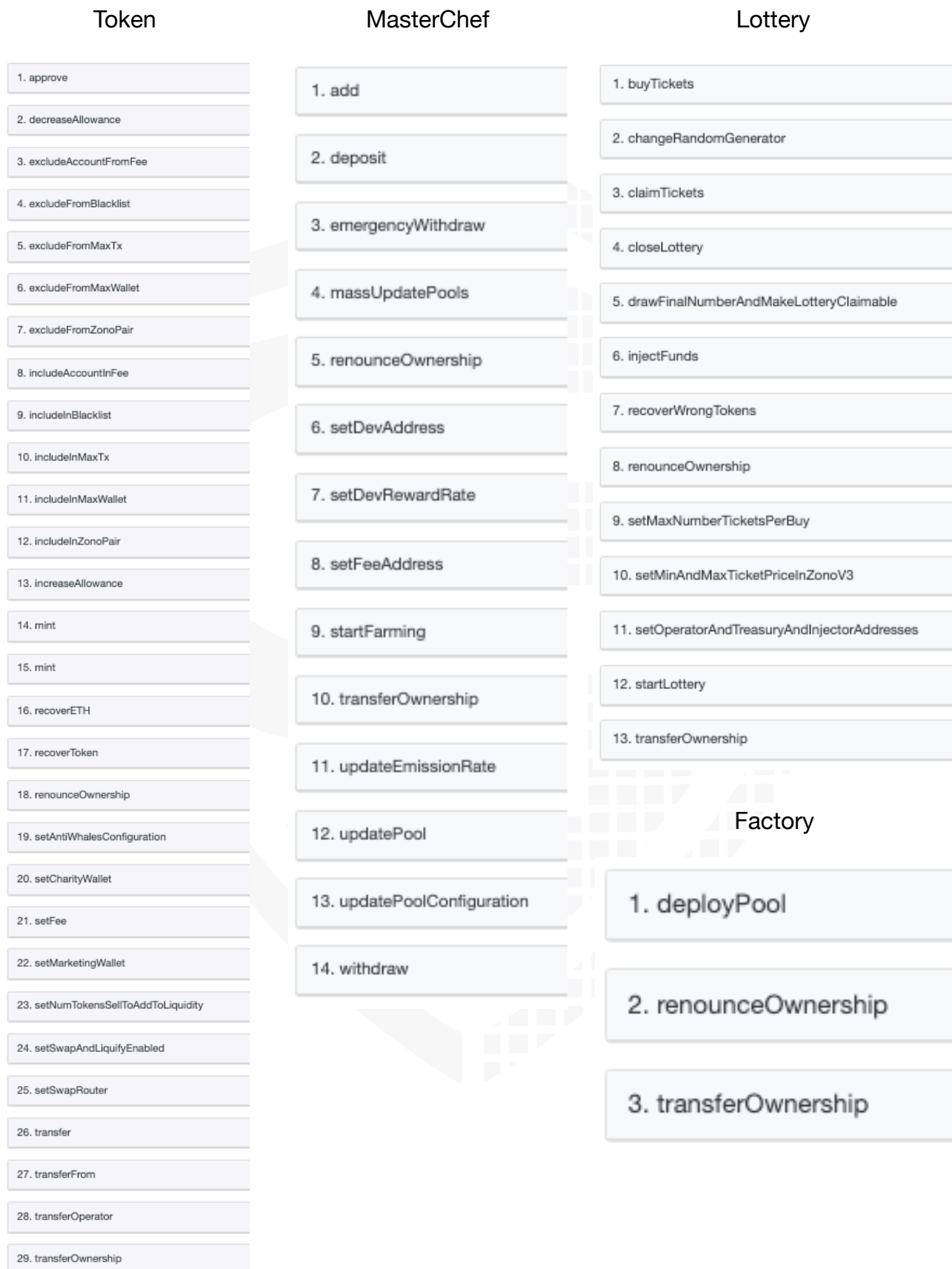
We will verify the following claims:
1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

## 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

## Token

1. approve
2. decreaseAllowance
3. excludeAccountFromFee
4. excludeFromBlacklist
5. excludeFromMaxTx
6. excludeFromMaxWallet
7. excludeFromZonoPair
8. includeAccountInFee
9. includeInBlacklist
10. includeInMaxTx
11. includeInMaxWallet
12. includeInZonoPair
13. increaseAllowance
14. mint
15. mint
16. recoverETH
17. recoverToken
18. renounceOwnership
19. setAntiWhalesConfiguration
20. setCharityWallet
21. setFee
22. setMarketingWallet
23. setNumTokensSellToAddToLiquidity
24. setSwapAndLiquifyEnabled
25. setSwapRouter
26. transfer
27. transferFrom
28. transferOperator
29. transferOwnership

## MasterChef

1. add
2. deposit
3. emergencyWithdraw
4. massUpdatePools
5. renounceOwnership
6. setDevAddress
7. setDevRewardRate
8. setFeeAddress
9. startFarming
10. transferOwnership
11. updateEmissionRate
12. updatePool
13. updatePoolConfiguration
14. withdraw

## Lottery

1. buyTickets
2. changeRandomGenerator
3. claimTickets
4. closeLottery
5. drawFinalNumberAndMakeLotteryClaimable
6. injectFunds
7. recoverWrongTokens
8. renounceOwnership
9. setMaxNumberTicketsPerBuy
10. setMinAndMaxTicketPriceInZonoV3
11. setOperatorAndTreasuryAndInjectorAddresses
12. startLottery
13. transferOwnership

## Factory

1. deployPool
2. renounceOwnership
3. transferOwnership

# 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 |
|-----------|--------|
| Verfified / Checked | ✓ |
| Partly Verified | 🚩 |
| Unverified / Not checked | ✗ |
| Not available | – |

# Modifiers and public functions
## v1.0

Token

MasterChef

**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

## Pool

### initialize

### deposit
- nonReentrant

### withdraw
- nonReentrant

### emergencyWithdraw
- nonReentrant

### emergencyRewardWithdraw
- onlyOwner

### recoverWrongTokens
- onlyOwner

### stopReward
- onlyOwner

### updatePoolLimitPerUser
- onlyOwner

### updateRewardPerBlock
- onlyOwner

### updateDepositFee
- onlyOwner

### updateFeeAddress
- onlyOwner

### updateStartAndEndBlocks
- onlyOwner

## Factory

### deployPool
- onlyOwner
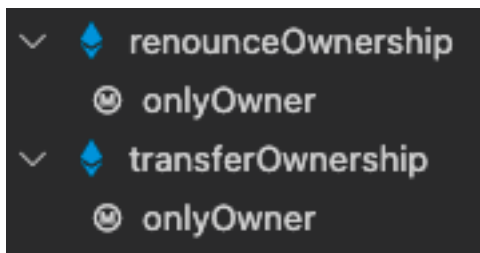
## Comments

- Deployer can set following state variables without any limitations
  - Token
    - _numTokensSellToAddToLiquidity

  - Lottery
    - maxNumberTicketsPerBuyOrClaim
    - minPriceTicketInZonoV3
    - maxPriceTicketInZonoV3

  - MasterChef
    - zonoPerBlock
  - FactoryPool
    - startBlock
    - bonusEndBlock
    - lastRewardBlock
    - rewardPerBlock
    - poolLimitPerUser
      - Can only be set if hasUserLimit is true. If it is set to false in updatePoolLimitPerUser function it cannot be set anymore because of the require statement which is looking for true value

- 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
- Factory can create new pools

**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/ZonoV3PoolFactory.sol | 8 | 1 | 1451 | 1210 | 597 | 562 | 355 | |
| | 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 | |
| | **Totals** | **31** | **10** | **6490** | **5358** | **2846** | **2308** | **1725** | |

## 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 | 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 | MasterChef | Missing Events Arithmetic | 1059, 1107, 1536 | Emit an event for critical parameter changes |

# Informational issues

| Issue | File | Type | Line | Description |
|---|---|---|---|---|
| #1 | MasterChef | State variables that could be declared constant (constable-states) | 900 | Add the `constant` attributes to state variables that never change |
| #2 | MasterChef | Unused state variables | 900 | Remove unused state variables |
| #3 | MasterChef | Misspelling | See description | Change following words:<br><br>- vairables L1179<br><br>Make sure to change it everywhere else as well. |
| #4 | Token | Misspelling | See description | Change following words:<br><br>- MarketingFeeTrasferred L1481<br>- recieve L1503<br>- swaping L1503<br><br>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. |
| #6 | Factory | Unnecessary require statement | 1419, 1420 | In the code is an unnecessary require statement because total supply is always over 0 |

# 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.

## 30. April 2022:

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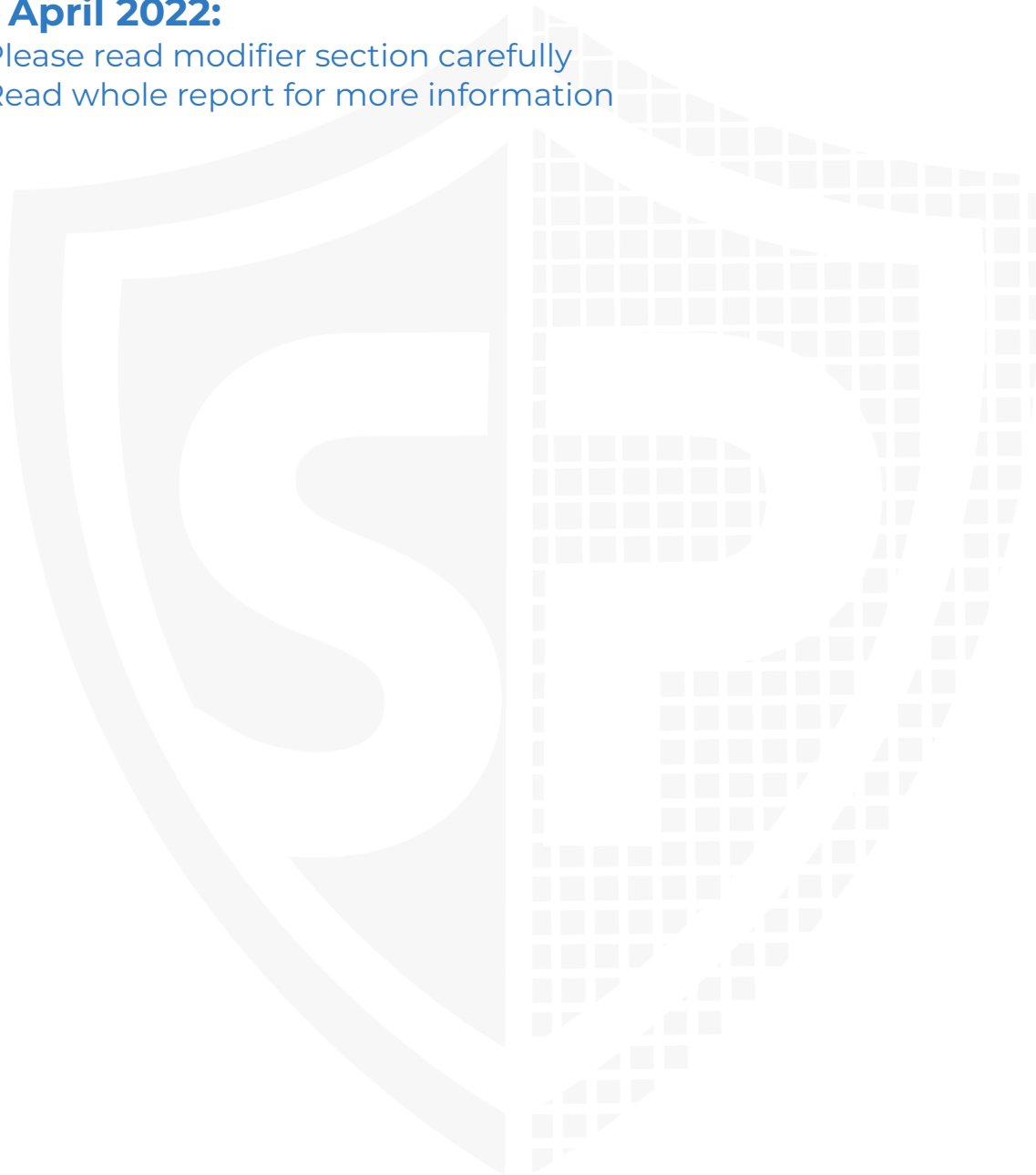
# SWC Attacks

| 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 | **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](#) | **NOT 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 | 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