Disclaimer

Version	Date	Description
1.0	07. July 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://winbinary.net/

Telegram
https://t.me/WinBinary01

Twitter
https://twitter.com/Win_Binary01

Discord
https://discord.gg/DVkbScDM

Description

WinBinary is a Binance Smart Chain Dividend Token protocol that rewards BUSD to its holders. The BUSD rewards are reflected from 10% buy and sell tax transactions and weekly manual reward pooled from all the listed utilities development revenue 50% share.

Project Engagement

During the 6th of July 2022, **WinBinary 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

- https://bscscan.com/address/ 0xd9075d050ca8905c6e14053c52a09244e3049124#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:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/proxy/Clones.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/utils/math/SafeMath.sol	2

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/winbinary.sol	a8cfb8a7b968a62b01aee03f011c111d4fab6a4c
contracts/IUniswapV2Factory.sol	9360ecb8c004c1a459a7f1d0e3a57e7d83b25383
contracts/BaseToken.sol	9a217daf9678b6bb4eb3fe6e038e4af886e5e89c
contracts/WinBinaryTokenDividendTracker.sol	0045cf23c70afcdb5bd5b67ad1d81bfa8aac8604
contracts/IUniswapV2Router02.sol	a1d006daf0a8b0f8e403fa37d073edf19034d40a

Metrics

Source Lines
v1.0
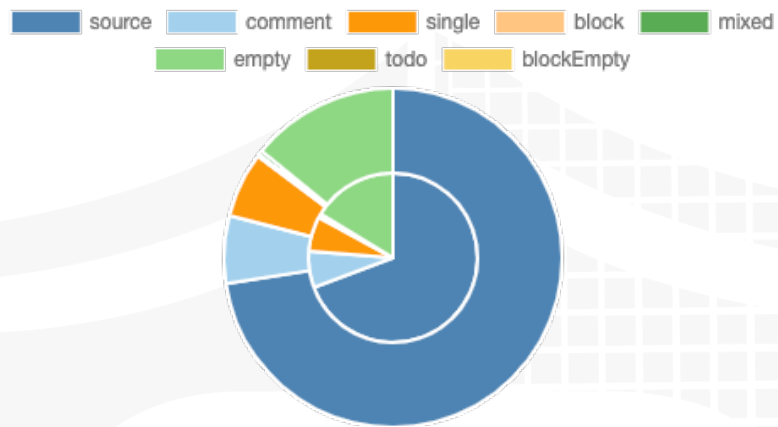
Risk Level
v1.0
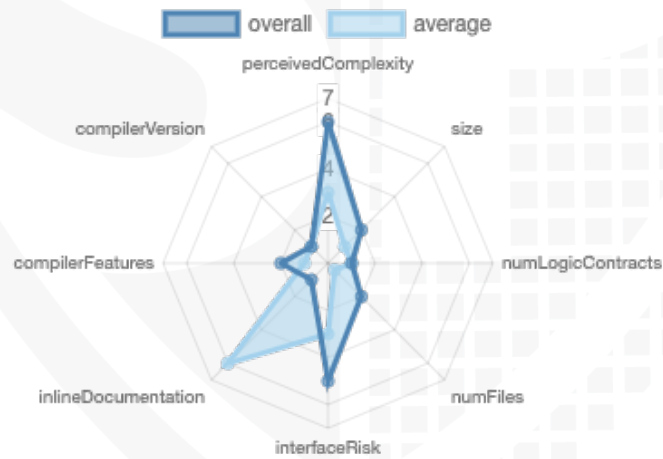
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	0	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	106	5

Version	External	Internal	Private	Pure	View
1.0	78	79	6	7	44

State Variables

Version	Total	Public
1.0	61	32

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`=0.8.4` `>=0.5.0` `>=0.6.2`		yes		

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

Inheritance Graph
v1.0
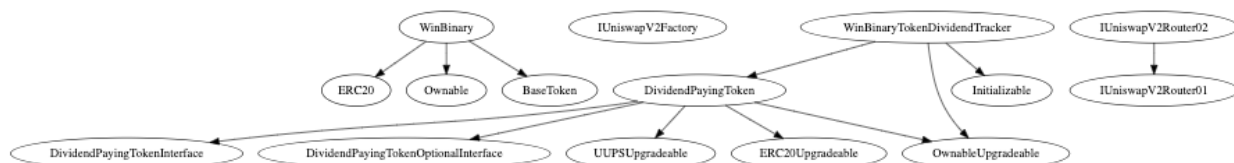
CallGraph
v1.0
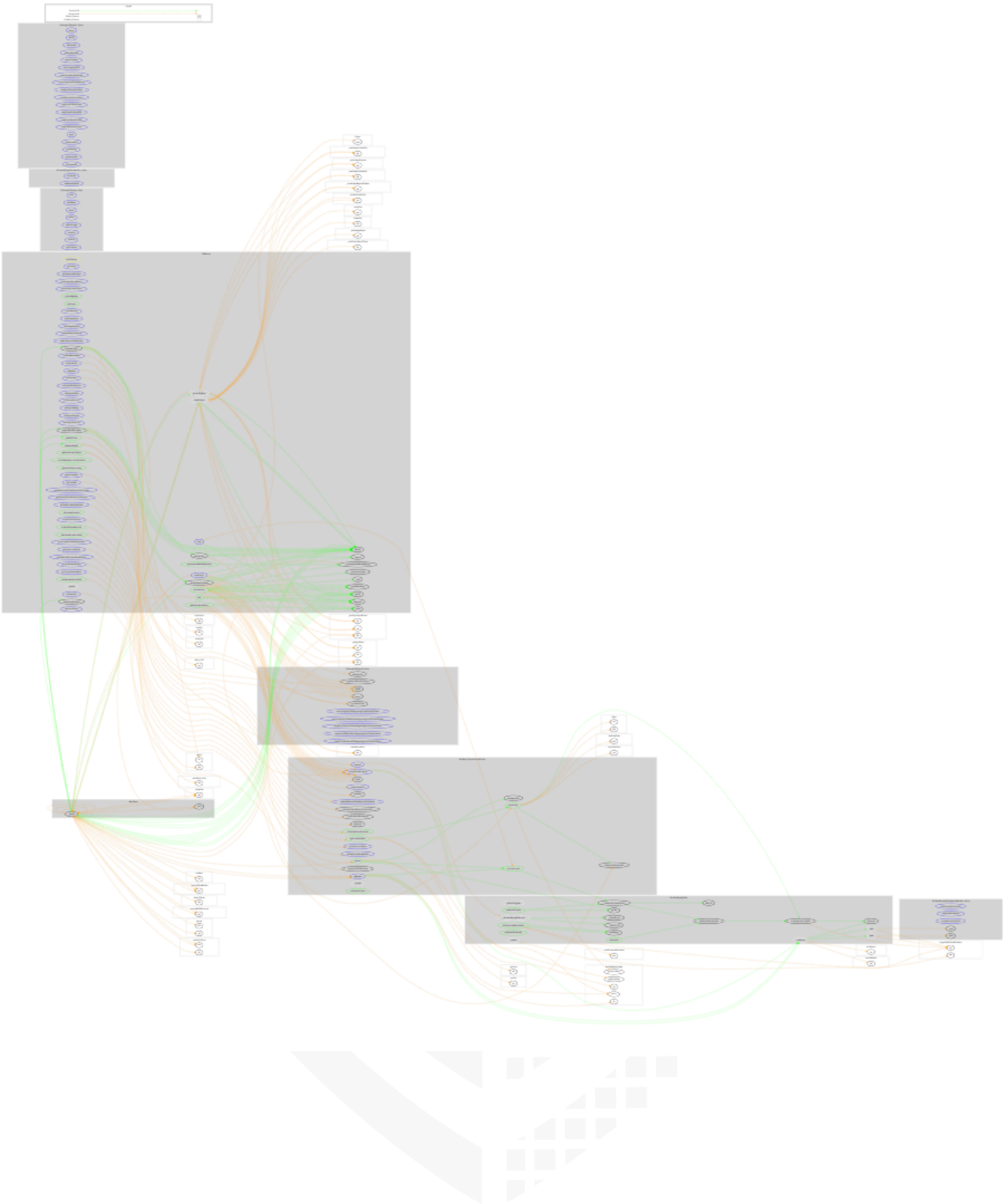
Scope of Work/Verify Claims

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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	**Yes**

Comments:
v1.0

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
    - Be aware of this and do your own research for the contract which is the contract pointing to

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

1. approve

2. burn

3. claim

4. decreaseAllowance

5. excludeFromDividends

6. excludeFromFees

7. excludeMultipleAccountsFromFees

8. fundJackpot

9. increaseAllowance

10. processDividendTracker

11. renounceOwnership

12. setAutomatedMarketMakerPair

13. setBurnFee

14. setBuyBackWallet

15. setJackpotFee

16. setJackpotHardBuyback

17. setJackpotHardLimit

18. setJackpotMinBuy

19. setJackpotTimespan

20. setLiquidityFee

21. setMaxJackpotLimitMultiplier

22. setSwapTokensAtAmount

23. setTokenRewardsFee

24. setTresuryFee

25. setTresuryReceiver

26. transfer

27. transferFrom

28. transferOwnership

29. updateClaimWait

30. updateDividendTracker

31. updateGasForProcessing

32. updateMinimumTokenBalanceForDividends

33. updateUniswapV2Router

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply			

Comments:
v1.0

- Owner can mint new tokens while _setBalance function

Deployer cannot burn or lock user funds

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

Comments:
v1.0
- Tokens
    - will be burned in _setBalance function
    - Will be burned while transfer
    - Can be burned by owner

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

Functions	Modifiers
setSwapTokensAtAmount	onlyOwner
setMaxJackpotLimitMultiplier	onlyOwner
setTokenRewardsFee	onlyOwner
setLiquidityFee	onlyOwner
setJackpotFee	onlyOwner
setBurnFee	onlyOwner
setTresuryFee	onlyOwner
setJackpotHardBuyback	onlyOwner
setBuyBackWallet	onlyOwner
setTresuryReceiver	onlyOwner
setJackpotMinBuy	onlyOwner
setJackpotTimespan	onlyOwner
setJackpotHardLimit	onlyOwner
fundJackpot	onlyOwner
updateDividendTracker	onlyOwner
updateUniswapV2Router	onlyOwner
excludeFromFees	onlyOwner
excludeMultipleAccountsFromF...	onlyOwner
setAutomatedMarketMakerPair	onlyOwner
updateGasForProcessing	onlyOwner
updateClaimWait	onlyOwner
updateMinimumTokenBalanceF...	onlyOwner
excludeFromDividends	onlyOwner
processDividendTracker	
claim	
burn	

Functions	Modifiers
initialize	
excludeFromDividends	onlyOwner
updateClaimWait	onlyOwner
updateMinimumTokenBala...	onlyOwner
setBalance	onlyOwner
process	
processAccount	onlyOwner

Comments
- *Deployer can set following state variables without any limitations*
  - _pendingJackpotBalance
  - _jackpotTokens
  - jackpotHardLimit
  - jackpotTimespan
  - jackpotMinBuy
  - jackpotHardBuyback
  - maxJackpotLimitMultiplier
  - swapTokensAtAmount
  - minimumTokenBalanceForDividends

- excludedFromDividends

- *Deployer can enable/disable following state variables*
  - automatedMarketMakerPairs
  - _isExcludedFromFees

- *Deployer can set following addresses*
  - uniswapV2Router
  - uniswapV2Pair
  - dividendTracker
  - tresuryReceiver
  - buybackWallet

- *Existing Modifiers*
  - lockTheSwap

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/winbinary.sol	1	—	969	869	685	42	565	💰♻️👥
🔍	contracts/IUniswapV2Pair.sol	—	1	111	12	9	1	55	—
📚	contracts/SafeMathUint.sol	1	—	14	14	8	5	3	—
🔍	contracts/IUniswapV2Factory.sol	—	1	32	12	9	1	17	—
🍃	contracts/BaseToken.sol	1	—	22	22	19	1	1	—
📚	contracts/SafeMathInt.sol	1	—	65	65	33	23	16	—
📝🔍	contracts/WinBinaryTokenDividendTracker.sol	2	2	552	417	281	79	192	♻️
📚	contracts/IterableMapping.sol	1	—	73	61	48	2	7	—
🔍	contracts/IUniswapV2Router02.sol	—	2	208	6	4	1	64	💰
📝📚🔍🍃	**Totals**	**7**	**6**	**2046**	**1478**	**1096**	**155**	**920**	💰♻️👥

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Interfaces	A floating pragma is set	At the top of files	Check the pragma version and use a specific one instead of floating pragmas.
#2	Main	Missing Zero Address Validation (missing-zero-check)	191 362 366 540	Check that the address is not zero
#3	Main	State variable visibility is not set	35	It is best practice to set the visibility of state variables explicitly
#4	WinBinaryTokenDividendTracker	Local variables shadowing	118 119 211 178 185 197	Rename the local variables that shadow another component

#5	Main	Missing Events Arithmetic	338 328 358 378 370 374 318 304 297 348 329-333 319-323 309-313 349-353	Emit an event for critical parameter changes
#6	WinBinaryTokenDividendTracker	Initializer modifier missing	312	Modifier is missing, that means you can call the initialize function again

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	44, 42	Add the `constant` attributes to state variables that never change
#2	SafeMathInt	Unused state variables	10	Remove unused state variables
#3	Main	Unused state variables	30	Remove unused state variables
#4	WinBinaryTokenDividendTracker	Error message is missing	127, 235, 341	Provide an error message for require statement
#5	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#6	WinBinaryTokenDividendTracker	Unnecessary code	237-244	Code is not reachable

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

07. July 2022:

- This contracts can be upgraded. Be aware of it
- Read whole report and modifiers section 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	**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](#)	**NOT PASSED**
[SWC-118](#)	Incorrect Constructor Name	[CWE-665: Improper Initialization](#)	**PASSED**
[SWC-117](#)	Signature Malleability	[CWE-347: Improper Verification of Cryptographic Signature](#)	**PASSED**

29

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**