Hash0X

SMART CONTRACT CODE SECURITY ANALYSIS REPORT

Project: JetSwap

Customer: Jetfuel Finance

Date: 21/04/2021

Table of Content

Disclaimer

Hash0X 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 or project that contracts Hash0X to perform a security review.

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The content of this audit report is provided “as is”, without representations and warranties of any kind, and Hash0X disclaims any liability for damage arising out of, or in connection with, this audit report. Copyright of this report remains with Hash0X.

Purpose of the report

The Audits and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the Solidity programming language that could present security risks. Cryptographic tokens and smart contracts are emergent technologies and carry with them high levels of technical risk and uncertainty.

The Audits are not an endorsement or indictment of any particular project or team, and the Audits do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset.

Introduction

We first thank Jetfuel Team for giving us the opportunity to audit their smart contract. This document outlines our methodology, audit details, and results. Jetfuel Team asked us to review their JetSwap Protocol smart contracts.

Hash0X reviewed the system from a technical perspective looking for bugs, issues and vulnerabilities in their code base. This audit report is valid for the smart contract at the mentioned commit hashes only. This audit report is not valid for any other versions of the smart contract files.

Project files

File Name	MD5 Hash
MasterChef.sol	67A621040EA4CD5B851BC8E797D87F8A
Multicall.sol	C5C1107C4FC647B326284AF5FD0B00EE
JetswapFactory.sol	A6440A04AC2D604CC79A4C62A8C89120
JetswapRouter.sol	1140683976AAD1D9A7796FC38957AF94
WingsToken.sol	EBCE5069F77A8D0F398133B03F5CD5B8

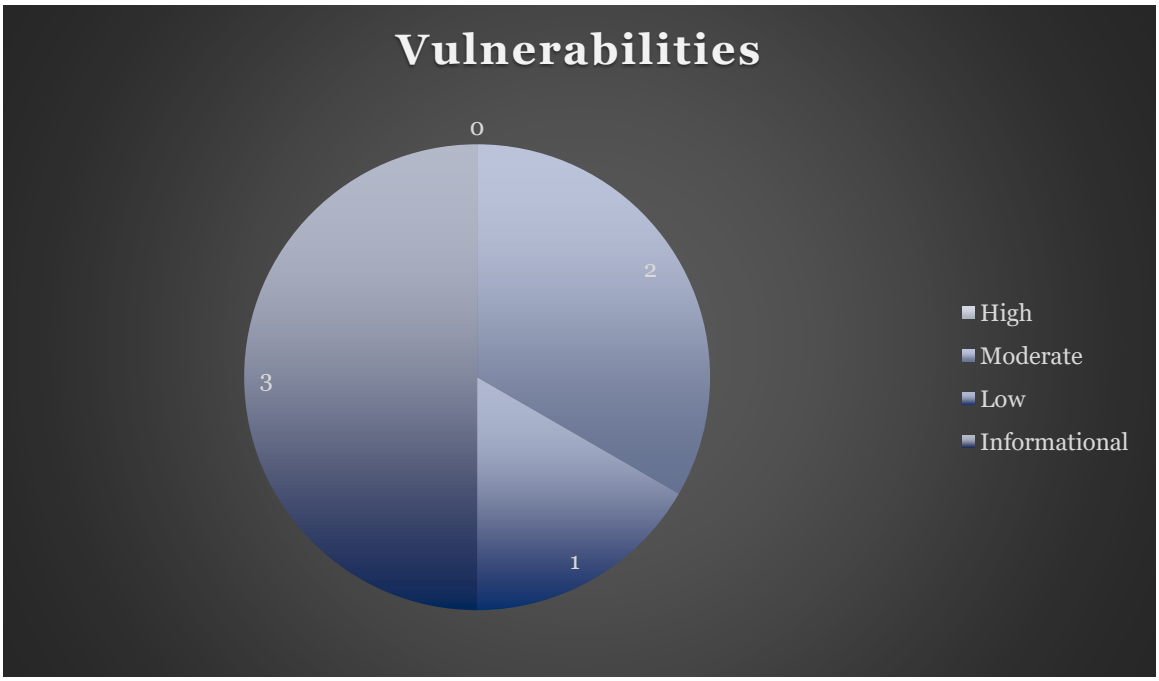
Audit Summary

Several issues were found in the first audit which were fixed by the Jetfuel Team and no issues were found in the revised audit of the patched version.

High Severity Issues Found	0
Moderate Severity Issues Found	2
Low Severity Issues	1
Informational Observations	3

The smart contract is considered to **pass the audit**, as of the audit date, if no high severity or moderate severity issues are found.

Finding Chart



These vulnerabilities were resolved by Jetfuel Team in the patched version of the smart contracts by taking necessary measures.

Overview

The project has 5 core Solidity files for the JetSwap Protocol Smart Contract, the JetSwap Protocol has smart contracts which interacts with each other in a complex way. We manually reviewed each line of code in the smart contract within the scope.

Methodology

Hash0X manually reviewed the smart contract line-by-line, keeping in mind industry best practices and known attacks, looking for any potential issues and vulnerabilities, and areas where improvements are possible.

We also used automated tools like slither / surya for analysis and reviewing the smart contract. These tools often give false-positives, and any issues reported by them but not included in the issue list can be considered not valid.

Classification / Issue Types Definition:

1. **High Severity**: which presents a significant security vulnerability or failure of the contract across a range of scenarios, or which may result in loss of funds.
2. **Moderate Severity**: which affects the desired outcome of the contract execution or introduces a weakness that can be exploited. It may not result in loss of funds but breaks the functionality or produces unexpected behavior.
3. **Low Severity**: which does not have a material impact on the contract execution and is likely to be subjective.

As mentioned above, the smart contract is considered to pass the audit, as of the audit date, if no high severity or moderate severity issues are found.

Attacks & Issues considered while auditing

In order to check for the security of the contract, we reviewed each line of code in the smart contract considering several known Smart Contract Attacks & known issues

- **Overflows and underflows**

  An overflow happens when the limit of the type variable uint256 , 2 ** 256, is exceeded. What happens is that the value resets to zero instead of incrementing more. For instance, if we want to assign a value to a uint bigger than 2 ** 256 it will simple go to 0—this is dangerous. On the other hand, an underflow happens when you try to subtract 0 minus a number bigger than 0. For example, if you subtract 0 - 1 the result will be = 2 ** 256 instead of -1. This is quite dangerous. This contract DOES check for overflows and underflows using SafeMath libraries.

- **Reentrancy Attack**

  One of the major dangers of calling external contracts is that they can take over the control flow, and make changes to your data that the calling function wasn't expecting. This class of bug can take many forms, and both of the major bugs that led to the DAO's collapse were bugs of this sort. This smart contract uses Check-effect pattern to protect against this attack.

- **Replay attack**

  The replay attack consists of making a transaction on one blockchain like the original Ethereum’s blockchain and then repeating it on another blockchain like the Ethereum’s classic blockchain. The ether is transferred like a normal transaction from a blockchain to another. Though it's no longer a problem because since the version 1.5.3 of Geth and 1.4.4 of Parity both implement the attack protection EIP 155 by Vitalik Buterin. So, the people that will use the contract depend on their own ability to be updated with those programs to keep themselves secure. Since this full system is a cross-chain bridge between Binance Smart Chain and Ethereum Blockchains – it is recommended to not let users enter arbitrary chain IDs in the transfer and receipt requests which may result in a potential replay attack in the future.

- **Short address attack**

  This attack affects ERC20 tokens, was discovered by the Golem team and consists of the following: A user creates an Ethereum wallet with a trailing 0, which is not hard because it's only a digit. For instance: 0xiofa8d97756as7df5sd8f75g8675ds8gsdg0 Then he buys tokens by removing the last zero: Buy 1000 tokens from account 0xiofa8d97756as7df5sd8f75g8675ds8gsdg. If the contract has enough amount of tokens and the buy function doesn't check the length of the address of the sender, the Ethereum's virtual machine will just add zeroes to the transaction until the address is complete. This issue is not applicable to JetSwap smart contracts.

- **Approval Double-spend**

  ERC20 Standard allows users to approve other users to manage their tokens, or spend tokens from their account till a certain amount, by setting the user's allowance with the standard `approve` function, then the allowed user may use `transferFrom` to spend the allowed tokens. Hypothetically, given a situation where Alice approves Bob to spend 100 Tokens from her account, and if Alice needs to adjust the allowance to allow Bob to spend 20 more tokens, normally – she'd check Bob's allowance (100 currently) and start a new `approve` transaction allowing Bob to spend a total of 120 Tokens instead of 100 Tokens.

  Likely impact of this bug is low for most situations. For more, see this discussion on GitHub: https://github.com/ethereum/EIPs/issues/20#issuecomment263524729

- **Sybil attacks**

  In a Sybil attack, the attacker subverts the reputation system of a network service by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence. Normally in BEP20 & DeFi smart contracts, sybil attacks are related to voting power amplification where a user may use their vote more than once in case of governance tokens. Proper user input restrictions in the governance contract prevented this type of attack.

Issues Found

High Severity Issues

No high severity issues were found in the smart contract.

Moderate Severity Issues

(1) Unlimited minting possibility

```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
}
```

In WingsToken.sol smart contract, we observed that the minting of tokens can be done unlimited by the owner. It is recommended to limit the token minting to prevent inflation of the token value.

Fix: Jetfuel Team confirmed that minting will be governed by the MasterChef smart contract, thus, preventing inflation of the token value.

(2) Function validation missing

```
// Add a new lp to the pool. Can only be called by the owner.
// XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
```

In MasterChef.sol contract, add function has logical requirement that the same LP tokens must not be added twice. This will create discrepancy in the logic.

Fix: Jetfuel Team confirmed that this will be taken care as add function is owner only function.

Low Severity Issues

(1) Possibility of infinite loop

```
function _swap(uint[] memory amounts, address[] memory path, address _to) internal virtual {
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0,) = JetswapLibrary.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
        address to = i < path.length - 2 ? JetswapLibrary.pairFor(factory, output, path[i + 2]) : _to;
        IJetswapPair(JetswapLibrary.pairFor(factory, input, output)).swap(
            amount0Out, amount1Out, to, new bytes(0)
        );
    }
}
```

As seen in smart contract functional summery section, there are many places where there are loops without any limits. It is recommended to put limits in the loops to prevent any scenarios where block's gas limit would hit.

Fix: we got confirmation from the Jetfuel Team that this would be taken care from the client side.

Informational Observations

(1) Event logs should be there in the function where state change is happening. For example:

a. setFeeTo function in JetswapFactory.sol
b. setFeeToSetter function in JetswapFactory.sol
c. initialize function in JetswapFactory.sol

(2) Use latest solidity version: It is recommended to use latest solidity version as they fix many compiler levels bugs from the old versions.

(3) Use visibility External over public: If any function is not being called internally, then it is better to specify its visibility as external. It saves some gas as well. https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Audit Conclusion

The JetSwap protocol smart contract codes were written very systematic, that we did not find any major issues in it. Hence, this code is ready for the production.

Due to the nature of the smart contract protocol, there are unlimited use case scenarios, thus it is not possible to give guarantee about the future outcomes. This audit is based on manual code analysis as well as used latest static tools.

This audit report presents all the findings based on standard audit procedure, which includes manual analysis as well as automated software tools. Smart Contract's high-level description of functions was presented in Appendix section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract based on extensive audit procedure scope is **"Well Secured"**.

Appendix

Smart Contract Functional Summary

MasterChef.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	updateMultiplier	write	Passed	No Issue
3	poolLength	read	Passed	No Issue
4	add	write	Input validation missing	LP Token must not be added twice
5	set	write	Passed	No Issue
6	updateStakingPool	internal	Infinite loop possibility	Array length must be limited
7	getMultiplier	read	Passed	No Issue
8	pendingWings	read	Passed	No Issue
9	massUpdatePools	write	Infinite loop possibility	Array length must be limited
11	updatePool	write	Passed	No Issue
12	deposit	write	Passed	No Issue
13	withdraw	write	Passed	No Issue
14	enterStaking	write	Passed	No Issue
15	leaveStaking	write	Passed	No Issue
16	emergencyWithdraw	write	Passed	No Issue
17	safeWingsTransfer	write	Passed	No Issue
18	dev	write	Passed	No Issue

Multicall.sol

Sl.	Function	Type	Observation	Conclusion
1	aggregate	read	Passed	No Issue
2	getEthBalance	read	Passed	No Issue
3	getBlockHash	read	Passed	No Issue
4	getLastBlockHash	read	Passed	No Issue
5	getCurrentBlockTimestamp	read	Passed	No Issue
6	getCurrentBlockDifficulty	read	Passed	No Issue
7	getCurrentBlockGasLimit	read	Passed	No Issue
8	getCurrentBlockCoinbase	read	Passed	No Issue

JetswapFactory.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allPairsLength	read	Passed	No Issue
3	createPair	write	Passed	No Issue
4	setFeeTo	write	Passed	No Issue
5	setFeeToSetter	write	Passed	No Issue

JetswapRouter.sol

Sl	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_addLiquidity	internal	Passed	No Issue
3	addLiquidity	write	Passed	No Issue
4	addLiquidityETH	write	Passed	No Issue
5	removeLiquidity	write	Passed	No Issue
6	removeLiquidityETH	write	Passed	No Issue
7	removeLiquidityWithPermit	write	Passed	No Issue
8	removeLiquidityETHWithPermit	write	Passed	No Issue
9	removeLiquidityETH SupportingFeeOnTransferTokens	write	Passed	No Issue
10	removeLiquidityETHWith PermitSupportingFeeOn TransferTokens	internal	Passed	No Issue
11	_swap	internal	Infinite loop possibility	Keep path limited
12	swapExactTokensForTokens	write	Passed	No Issue
13	swapTokensForExactTokens	write	Passed	No Issue
14	swapExactETHForTokens	write	Passed	No Issue
15	swapTokensForExactETH	write	Passed	No Issue
16	swapExactTokensForETH	write	Passed	No Issue
17	swapETHForExactTokens	write	Passed	No Issue
18	_swapSupportingFeeOn TransferTokens	internal	Infinite loop possibility	Keep path limited
19	swapExactTokensForTokens SupportingFeeOnTransferTokens	write	Passed	No Issue
20	swapExactETHForTokens SupportingFeeOnTransferTokens	write	Passed	No Issue
21	swapExactTokensForETH SupportingFeeOnTransferTokens	write	Passed	No Issue
22	quote	read	Passed	No Issue
23	getAmountOut	read	Passed	No Issue
24	getAmountIn	read	Passed	No Issue
25	getAmountsOut	read	Passed	No Issue
26	getAmountsIn	read	Passed	No Issue

WingsToken.sol

Sl.	Function	Type	Observation	Conclusion
1	mint	write	No max minting set	must be used by masterchef contract
2	delegates	read	Passed	No Issue
3	delegate	write	Passed	No Issue
4	delegateBySig	write	Passed	No Issue
5	getCurrentVotes	read	Passed	No Issue
6	getPriorVotes	read	Infinite loop possibility	Keep array length limited
7	_delegate	internal	Passed	No Issue
8	_moveDelegates	internal	Passed	No Issue
9	_writeCheckpoint	internal	Passed	No Issue
10	safe32	read	Passed	No Issue
11	getChainId	read	Passed	No Issue

Slither Results Log

fedrik@fedrik-ThinkPad-T410:~/reports$ slither /home/fedrik/reports/MasterChef.sol
Compilation warnings/errors on /home/fedrik/reports/MasterChef.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDXLicense-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> /home/fedrik/reports/MasterChef.sol
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDXLicense-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> MasterChef.sol
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.
--> /home/fedrik/reports/MasterChef.sol:967:5:
|
967 | /// @notice A record of each accounts delegate
| ^^^
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.
--> /home/fedrik/reports/MasterChef.sol:1231:5:
|
1231 | /// @notice A record of each accounts delegate
| ^^^
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.
--> MasterChef.sol:967:5:
|
967 | /// @notice A record of each accounts delegate
| ^^^
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.
--> MasterChef.sol:1231:5:
|
1231 | /// @notice A record of each accounts delegate
| ^^^
INFO:Detectors:
MasterChef.pendingWings(uint256,address) (MasterChef.sol#1598-1609) performs a multiplication on the result of a division:
-wingsReward = multiplier.mul(wingsPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1605)
-accWingsPerShare = accWingsPerShare.add(wingsReward.mul(1e12).div(lpSupply))

(MasterChef.sol#1606)
MasterChef.updatePool(uint256) (MasterChef.sol#1621-1637) performs a multiplication on the result of a division:
-wingsReward = multiplier.mul(wingsPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1632)
-pool.accWingsPerShare = pool.accWingsPerShare.add(wingsReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
WingsToken._writeCheckpoint(address,uint32,uint256,uint256) (MasterChef.sol#1160-1178) uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (MasterChef.sol#1170)
SyrupBar._writeCheckpoint(address,uint32,uint256,uint256) (MasterChef.sol#1424-1442) uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (MasterChef.sol#1434)
MasterChef.updatePool(uint256) (MasterChef.sol#1621-1637) uses a dangerous strict equality:
- lpSupply == 0 (MasterChef.sol#1627)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strictequalities
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1551-1564):
External calls:
- massUpdatePools() (MasterChef.sol#1553)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
State variables written after the call(s):
- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0)) (MasterChef.sol#1557-1562)
- updateStakingPool() (MasterChef.sol#1563)
- poolInfo[0].allocPoint = points (MasterChef.sol#1588)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterChef.sol#1556)
- updateStakingPool() (MasterChef.sol#1563)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (MasterChef.sol#1587)
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1640-1659):
External calls:
- updatePool(_pid) (MasterChef.sol#1646)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1650)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1654)
State variables written after the call(s):

- user.amount = user.amount.add(_amount) (MasterChef.sol#1655)
- user.rewardDebt = user.amount.mul(pool.accWingsPerShare).div(1e12)
(MasterChef.sol#1657)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1724-1731):
External calls:
- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1727)
State variables written after the call(s):
- user.amount = 0 (MasterChef.sol#1729)
- user.rewardDebt = 0 (MasterChef.sol#1730)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1683-1701):
External calls:
- updatePool(0) (MasterChef.sol#1686)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1690)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(MasterChef.sol#1694)
State variables written after the call(s):
- user.amount = user.amount.add(_amount) (MasterChef.sol#1695)
- user.rewardDebt = user.amount.mul(pool.accWingsPerShare).div(1e12)
(MasterChef.sol#1697)
Reentrancy in MasterChef.leaveStaking(uint256) (MasterChef.sol#1704-1721):
External calls:
- updatePool(0) (MasterChef.sol#1708)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1711)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
State variables written after the call(s):
- user.amount = user.amount.sub(_amount) (MasterChef.sol#1714)
Reentrancy in MasterChef.leaveStaking(uint256) (MasterChef.sol#1704-1721):
External calls:
- updatePool(0) (MasterChef.sol#1708)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1711)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1715)
State variables written after the call(s):
- user.rewardDebt = user.amount.mul(pool.accWingsPerShare).div(1e12)
(MasterChef.sol#1717)
Reentrancy in MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1567-1577):
External calls:
- massUpdatePools() (MasterChef.sol#1569)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)

- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
State variables written after the call(s):
- poolInfo[_pid].allocPoint = _allocPoint (MasterChef.sol#1573)
- updateStakingPool() (MasterChef.sol#1575)
- poolInfo[0].allocPoint = points (MasterChef.sol#1588)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
(MasterChef.sol#1571)
- updateStakingPool() (MasterChef.sol#1575)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points)
(MasterChef.sol#1587)
Reentrancy in MasterChef.updatePool(uint256) (MasterChef.sol#1621-1637):
External calls:
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
State variables written after the call(s):
- pool.accWingsPerShare =
pool.accWingsPerShare.add(wingsReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1635)
- pool.lastRewardBlock = block.number (MasterChef.sol#1636)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1662-1680):
External calls:
- updatePool(_pid) (MasterChef.sol#1669)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1672)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
State variables written after the call(s):
- user.amount = user.amount.sub(_amount) (MasterChef.sol#1675)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1662-1680):
External calls:
- updatePool(_pid) (MasterChef.sol#1669)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1672)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1676)
State variables written after the call(s):
- user.rewardDebt = user.amount.mul(pool.accWingsPerShare).div(1e12)
(MasterChef.sol#1678)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancyvulnerabilities-
1
INFO:Detectors:
SyrupBar.safeWingsTransfer(address,uint256) (MasterChef.sol#1216-1223) ignores return value by
wings.transfer(_to,wingsBal) (MasterChef.sol#1219)
SyrupBar.safeWingsTransfer(address,uint256) (MasterChef.sol#1216-1223) ignores return value by
wings.transfer(_to,_amount) (MasterChef.sol#1221)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BEP20.constructor(string,string).name (MasterChef.sol#690) shadows:
- BEP20.name() (MasterChef.sol#706-708) (function)
- IBEP20.name() (MasterChef.sol#214) (function)
BEP20.constructor(string,string).symbol (MasterChef.sol#690) shadows:
- BEP20.symbol() (MasterChef.sol#720-722) (function)
- IBEP20.symbol() (MasterChef.sol#209) (function)
BEP20.allowance(address,address).owner (MasterChef.sol#754) shadows:
- Ownable.owner() (MasterChef.sol#600-602) (function)
BEP20._approve(address,address,uint256).owner (MasterChef.sol#926) shadows:
- Ownable.owner() (MasterChef.sol#600-602) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MasterChef.constructor(WingsToken,SyrupBar,address,uint256,uint256)._devaddr
(MasterChef.sol#1519) lacks a zero-check on :
- devaddr = _devaddr (MasterChef.sol#1525)
MasterChef.dev(address)._devaddr (MasterChef.sol#1739) lacks a zero-check on :
- devaddr = _devaddr (MasterChef.sol#1741)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
addressvalidation
INFO:Detectors:
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1640-1659):
External calls:
- updatePool(_pid) (MasterChef.sol#1646)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1650)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(MasterChef.sol#1654)
Event emitted after the call(s):
- Deposit(msg.sender,_pid,_amount) (MasterChef.sol#1658)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1724-1731):
External calls:
- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1727)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,user.amount) (MasterChef.sol#1728)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1683-1701):
External calls:
- updatePool(0) (MasterChef.sol#1686)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1690)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)

(MasterChef.sol#1694)
- syrup.mint(msg.sender,_amount) (MasterChef.sol#1699)
Event emitted after the call(s):
- Deposit(msg.sender,0,_amount) (MasterChef.sol#1700)
Reentrancy in MasterChef.leaveStaking(uint256) (MasterChef.sol#1704-1721):
External calls:
- updatePool(0) (MasterChef.sol#1708)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1711)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1715)
- syrup.burn(msg.sender,_amount) (MasterChef.sol#1719)
Event emitted after the call(s):
- Withdraw(msg.sender,0,_amount) (MasterChef.sol#1720)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1662-1680):
External calls:
- updatePool(_pid) (MasterChef.sol#1669)
- wings.mint(devaddr,wingsReward.div(10)) (MasterChef.sol#1633)
- wings.mint(address(syrup),wingsReward) (MasterChef.sol#1634)
- safeWingsTransfer(msg.sender,pending) (MasterChef.sol#1672)
- syrup.safeWingsTransfer(_to,_amount) (MasterChef.sol#1735)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1676)
Event emitted after the call(s):
- Withdraw(msg.sender,_pid,_amount) (MasterChef.sol#1679)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancyvulnerabilities-
3
INFO:Detectors:
WingsToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1026-
1067) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,WINGS::delegateBySig: signature expired)
(MasterChef.sol#1065)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1290-
1331) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,WINGS::delegateBySig: signature expired)
(MasterChef.sol#1329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MasterChef.sol#312-323) uses assembly
- INLINE ASM (MasterChef.sol#319-321)
Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#420-446) uses
assembly
- INLINE ASM (MasterChef.sol#438-441)

WingsToken.getChainId() (MasterChef.sol#1185-1189) uses assembly
- INLINE ASM (MasterChef.sol#1187)
SyrupBar.getChainId() (MasterChef.sol#1449-1453) uses assembly
- INLINE ASM (MasterChef.sol#1451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterChef.sol#341-347):
- (success) = recipient.call{value: amount}() (MasterChef.sol#345)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string)
(MasterChef.sol#420-446):
- (success,returndata) = target.call{value: weiValue}(data) (MasterChef.sol#429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter WingsToken.mint(address,uint256)._to (MasterChef.sol#956) is not in mixedCase
Parameter WingsToken.mint(address,uint256)._amount (MasterChef.sol#956) is not in mixedCase
Variable WingsToken._delegates (MasterChef.sol#968) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (MasterChef.sol#1195) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (MasterChef.sol#1195) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (MasterChef.sol#1200) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (MasterChef.sol#1200) is not in mixedCase
Parameter SyrupBar.safeWingsTransfer(address,uint256)._to (MasterChef.sol#1216) is not in
mixedCase
Parameter SyrupBar.safeWingsTransfer(address,uint256)._amount (MasterChef.sol#1216) is not in
mixedCase
Variable SyrupBar._delegates (MasterChef.sol#1232) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._allocPoint (MasterChef.sol#1551) is not in
mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._lpToken (MasterChef.sol#1551) is not in
mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._withUpdate (MasterChef.sol#1551) is not in
mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._pid (MasterChef.sol#1567) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._allocPoint (MasterChef.sol#1567) is not in
mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._withUpdate (MasterChef.sol#1567) is not in
mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._from (MasterChef.sol#1593) is not in
mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (MasterChef.sol#1593) is not in
mixedCase
Parameter MasterChef.pendingWings(uint256,address)._pid (MasterChef.sol#1598) is not in
mixedCase
Parameter MasterChef.pendingWings(uint256,address)._user (MasterChef.sol#1598) is not in
mixedCase
Parameter MasterChef.updatePool(uint256)._pid (MasterChef.sol#1621) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._pid (MasterChef.sol#1640) is not in mixedCase

Parameter MasterChef.deposit(uint256,uint256)._amount (MasterChef.sol#1640) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (MasterChef.sol#1662) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (MasterChef.sol#1662) is not in mixedCase
Parameter MasterChef.enterStaking(uint256)._amount (MasterChef.sol#1683) is not in mixedCase
Parameter MasterChef.leaveStaking(uint256)._amount (MasterChef.sol#1704) is not in mixedCase
Parameter MasterChef.emergencyWithdraw(uint256)._pid (MasterChef.sol#1724) is not in mixedCase
Parameter MasterChef.safeWingsTransfer(address,uint256)._to (MasterChef.sol#1734) is not in mixedCase
Parameter MasterChef.safeWingsTransfer(address,uint256)._amount (MasterChef.sol#1734) is not in mixedCase
Parameter MasterChef.dev(address)._devaddr (MasterChef.sol#1739) is not in mixedCase
Variable MasterChef.BONUS_MULTIPLIER (MasterChef.sol#1501) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soliditynaming-
conventions
INFO:Detectors:
Redundant expression "this (MasterChef.sol#565)" inContext (MasterChef.sol#555-568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (MasterChef.sol#619-622)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (MasterChef.sol#628-630)
decimals() should be declared external:
- BEP20.decimals() (MasterChef.sol#713-715)
symbol() should be declared external:
- BEP20.symbol() (MasterChef.sol#720-722)
totalSupply() should be declared external:
- BEP20.totalSupply() (MasterChef.sol#727-729)
transfer(address,uint256) should be declared external:
- BEP20.transfer(address,uint256) (MasterChef.sol#746-749)
allowance(address,address) should be declared external:
- BEP20.allowance(address,address) (MasterChef.sol#754-756)
approve(address,uint256) should be declared external:
- BEP20.approve(address,uint256) (MasterChef.sol#765-768)
transferFrom(address,address,uint256) should be declared external:
- BEP20.transferFrom(address,address,uint256) (MasterChef.sol#782-794)
increaseAllowance(address,uint256) should be declared external:
- BEP20.increaseAllowance(address,uint256) (MasterChef.sol#808-811)
decreaseAllowance(address,uint256) should be declared external:
- BEP20.decreaseAllowance(address,uint256) (MasterChef.sol#827-834)
mint(uint256) should be declared external:
- BEP20.mint(uint256) (MasterChef.sol#844-847)

mint(address,uint256) should be declared external:
- WingsToken.mint(address,uint256) (MasterChef.sol#956-959)
mint(address,uint256) should be declared external:
- SyrupBar.mint(address,uint256) (MasterChef.sol#1195-1198)
burn(address,uint256) should be declared external:
- SyrupBar.burn(address,uint256) (MasterChef.sol#1200-1203)
safeWingsTransfer(address,uint256) should be declared external:
- SyrupBar.safeWingsTransfer(address,uint256) (MasterChef.sol#1216-1223)
updateMultiplier(uint256) should be declared external:
- MasterChef.updateMultiplier(uint256) (MasterChef.sol#1541-1543)
add(uint256,IBEP20,bool) should be declared external:
- MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1551-1564)
set(uint256,uint256,bool) should be declared external:
- MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1567-1577)
deposit(uint256,uint256) should be declared external:
- MasterChef.deposit(uint256,uint256) (MasterChef.sol#1640-1659)
withdraw(uint256,uint256) should be declared external:
- MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1662-1680)
enterStaking(uint256) should be declared external:
- MasterChef.enterStaking(uint256) (MasterChef.sol#1683-1701)
leaveStaking(uint256) should be declared external:
- MasterChef.leaveStaking(uint256) (MasterChef.sol#1704-1721)
emergencyWithdraw(uint256) should be declared external:
- MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1724-1731)
dev(address) should be declared external:
- MasterChef.dev(address) (MasterChef.sol#1739-1742)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-thatcould-
be-declared-external
INFO:Slither:/home/fedrik/reports/MasterChef.sol analyzed (10 contracts with 72 detectors), 94 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
fedrik@fedrik-ThinkPad-T410:~/reports$
fedrik@fedrik-ThinkPad-T410:~/reports$ slither /home/fedrik/reports/Multicall.sol
Compilation warnings/errors on /home/fedrik/reports/Multicall.sol:
/home/fedrik/reports/Multicall.sol:6:1: Warning: Experimental features are turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
Multicall.sol:6:1: Warning: Experimental features are turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^------------------------------^
INFO:Detectors:
Multicall.aggregate(Multicall.Call[]) (Multicall.sol#19-27) has external calls inside a loop: (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#23)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version>=0.5.0 (Multicall.sol#5) allows old versions
solc-0.5.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-ofsolidity
INFO:Detectors:
Low level call in Multicall.aggregate(Multicall.Call[]) (Multicall.sol#19-27):
- (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
aggregate(Multicall.Call[]) should be declared external:
- Multicall.aggregate(Multicall.Call[]) (Multicall.sol#19-27)
getEthBalance(address) should be declared external:
- Multicall.getEthBalance(address) (Multicall.sol#29-31)
getBlockHash(uint256) should be declared external:
- Multicall.getBlockHash(uint256) (Multicall.sol#32-34)
getLastBlockHash() should be declared external:
- Multicall.getLastBlockHash() (Multicall.sol#35-37)
getCurrentBlockTimestamp() should be declared external:
- Multicall.getCurrentBlockTimestamp() (Multicall.sol#38-40)
getCurrentBlockDifficulty() should be declared external:
- Multicall.getCurrentBlockDifficulty() (Multicall.sol#41-43)
getCurrentBlockGasLimit() should be declared external:
- Multicall.getCurrentBlockGasLimit() (Multicall.sol#44-46)
getCurrentBlockCoinbase() should be declared external:
- Multicall.getCurrentBlockCoinbase() (Multicall.sol#47-49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-thatcould-
be-declared-external
INFO:Slither:/home/fedrik/reports/Multicall.sol analyzed (1 contracts with 72 detectors), 12 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
fedrik@fedrik-ThinkPad-T410:~/reports$
fedrik@fedrik-ThinkPad-T410:~/reports$ slither /home/fedrik/reports/JetswapFactory.sol
INFO:Detectors:
JetswapPair._update(uint256,uint256,uint112,uint112) (JetswapFactory.sol#321-334) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (JetswapFactory.sol#323)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
JetswapPair._safeTransfer(address,address,uint256) (JetswapFactory.sol#292-295) uses a dangerous strict equality:
- require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),Jetswap: TRANSFER_FAILED) (JetswapFactory.sol#294)
JetswapPair.mint(address) (JetswapFactory.sol#358-379) uses a dangerous strict equality:
- _totalSupply == 0 (JetswapFactory.sol#367)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strictequalities
INFO:Detectors:
Reentrancy in JetswapPair.burn(address) (JetswapFactory.sol#382-404):
External calls:
- _safeTransfer(_token0,to,amount0) (JetswapFactory.sol#396)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1) (JetswapFactory.sol#397)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
State variables written after the call(s):
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
- blockTimestampLast = blockTimestamp (JetswapFactory.sol#332)
- kLast = uint256(reserve0).mul(reserve1) (JetswapFactory.sol#402)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
- reserve0 = uint112(balance0) (JetswapFactory.sol#330)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
- reserve1 = uint112(balance1) (JetswapFactory.sol#331)
Reentrancy in JetswapFactory.createPair(address,address) (JetswapFactory.sol#470-485):
External calls:
- IJetswapPair(pair).initialize(token0,token1) (JetswapFactory.sol#480)
State variables written after the call(s):
- getPair[token0][token1] = pair (JetswapFactory.sol#481)
- getPair[token1][token0] = pair (JetswapFactory.sol#482)
Reentrancy in JetswapPair.swap(uint256,uint256,address,bytes) (JetswapFactory.sol#407-435):
External calls:
- _safeTransfer(_token0,to,amount0Out) (JetswapFactory.sol#418)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1Out) (JetswapFactory.sol#419)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- IJetswapCallee(to).jetswapCall(msg.sender,amount0Out,amount1Out,data) (JetswapFactory.sol#420)
State variables written after the call(s):
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)
- blockTimestampLast = blockTimestamp (JetswapFactory.sol#332)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)
- reserve0 = uint112(balance0) (JetswapFactory.sol#330)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)
- reserve1 = uint112(balance1) (JetswapFactory.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancyvulnerabilities-1
INFO:Detectors:

JetswapPair.initialize(address,address)._token0 (JetswapFactory.sol#314) lacks a zero-check on :
- token0 = _token0 (JetswapFactory.sol#316)
JetswapPair.initialize(address,address)._token1 (JetswapFactory.sol#314) lacks a zero-check on :
- token1 = _token1 (JetswapFactory.sol#317)
JetswapFactory.constructor(address)._feeToSetter (JetswapFactory.sol#462) lacks a zero-check on :
- feeToSetter = _feeToSetter (JetswapFactory.sol#463)
JetswapFactory.setFeeTo(address)._feeTo (JetswapFactory.sol#487) lacks a zero-check on :
- feeTo = _feeTo (JetswapFactory.sol#489)
JetswapFactory.setFeeToSetter(address)._feeToSetter (JetswapFactory.sol#492) lacks a zero-check on :
- feeToSetter = _feeToSetter (JetswapFactory.sol#494)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-addressvalidation
INFO:Detectors:
Reentrancy in JetswapPair.burn(address) (JetswapFactory.sol#382-404):
External calls:
- _safeTransfer(_token0,to,amount0) (JetswapFactory.sol#396)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1) (JetswapFactory.sol#397)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
State variables written after the call(s):
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
- price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (JetswapFactory.sol#327)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
- price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (JetswapFactory.sol#328)
Reentrancy in JetswapFactory.createPair(address,address) (JetswapFactory.sol#470-485):
External calls:
- IJetswapPair(pair).initialize(token0,token1) (JetswapFactory.sol#480)
State variables written after the call(s):
- allPairs.push(pair) (JetswapFactory.sol#483)
Reentrancy in JetswapPair.swap(uint256,uint256,address,bytes) (JetswapFactory.sol#407-435):
External calls:
- _safeTransfer(_token0,to,amount0Out) (JetswapFactory.sol#418)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1Out) (JetswapFactory.sol#419)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (JetswapFactory.sol#293)
- IJetswapCallee(to).jetswapCall(msg.sender,amount0Out,amount1Out,data) (JetswapFactory.sol#420)
State variables written after the call(s):
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)

- price0CumulativeLast +=
uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (JetswapFactory.sol#327)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)
- price1CumulativeLast +=
uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (JetswapFactory.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancyvulnerabilities-
2
INFO:Detectors:
Reentrancy in JetswapPair.burn(address) (JetswapFactory.sol#382-404):
External calls:
- _safeTransfer(_token0,to,amount0) (JetswapFactory.sol#396)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1) (JetswapFactory.sol#397)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(JetswapFactory.sol#293)
Event emitted after the call(s):
- Burn(msg.sender,amount0,amount1,to) (JetswapFactory.sol#403)
- Sync(reserve0,reserve1) (JetswapFactory.sol#333)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#401)
Reentrancy in JetswapFactory.createPair(address,address) (JetswapFactory.sol#470-485):
External calls:
- IJetswapPair(pair).initialize(token0,token1) (JetswapFactory.sol#480)
Event emitted after the call(s):
- PairCreated(token0,token1,pair,allPairs.length) (JetswapFactory.sol#484)
Reentrancy in JetswapPair.swap(uint256,uint256,address,bytes) (JetswapFactory.sol#407-435):
External calls:
- _safeTransfer(_token0,to,amount0Out) (JetswapFactory.sol#418)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(JetswapFactory.sol#293)
- _safeTransfer(_token1,to,amount1Out) (JetswapFactory.sol#419)
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(JetswapFactory.sol#293)
- IJetswapCallee(to).jetswapCall(msg.sender,amount0Out,amount1Out,data)
(JetswapFactory.sol#420)
Event emitted after the call(s):
- Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to)
(JetswapFactory.sol#434)
- Sync(reserve0,reserve1) (JetswapFactory.sol#333)
- _update(balance0,balance1,_reserve0,_reserve1) (JetswapFactory.sol#433)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancyvulnerabilities-
3
INFO:Detectors:
JetswapERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32)

(JetswapFactory.sol#186-198) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(deadline >= block.timestamp,Jetswap: EXPIRED)
(JetswapFactory.sol#187)
JetswapPair._update(uint256,uint256,uint112,uint112) (JetswapFactory.sol#321-334) uses timestamp for comparisons
Dangerous comparisons:
- timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (JetswapFactory.sol#325)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
JetswapERC20.constructor() (JetswapFactory.sol#129-143) uses assembly
- INLINE ASM (JetswapFactory.sol#131-133)
JetswapFactory.createPair(address,address) (JetswapFactory.sol#470-485) uses assembly
- INLINE ASM (JetswapFactory.sol#477-479)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Low level call in JetswapPair._safeTransfer(address,address,uint256) (JetswapFactory.sol#292-295):
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(JetswapFactory.sol#293)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IJetswapPair.DOMAIN_SEPARATOR() (JetswapFactory.sol#38) is not in mixedCase
Function IJetswapPair.PERMIT_TYPEHASH() (JetswapFactory.sol#39) is not in mixedCase
Function IJetswapPair.MINIMUM_LIQUIDITY() (JetswapFactory.sol#56) is not in mixedCase
Function IJetswapERC20.DOMAIN_SEPARATOR() (JetswapFactory.sol#89) is not in mixedCase
Function IJetswapERC20.PERMIT_TYPEHASH() (JetswapFactory.sol#90) is not in mixedCase
Variable JetswapERC20.DOMAIN_SEPARATOR (JetswapFactory.sol#121) is not in mixedCase
Parameter JetswapPair.initialize(address,address)._token0 (JetswapFactory.sol#314) is not in mixedCase
Parameter JetswapPair.initialize(address,address)._token1 (JetswapFactory.sol#314) is not in mixedCase
Parameter JetswapFactory.setFeeTo(address)._feeTo (JetswapFactory.sol#487) is not in mixedCase
Parameter JetswapFactory.setFeeToSetter(address)._feeToSetter (JetswapFactory.sol#492) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soliditynaming-
conventions
INFO:Detectors:
Variable JetswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (JetswapFactory.sol#428) is too similar to JetswapPair.swap(uint256,uint256,address,bytes).balance1Adjusted (JetswapFactory.sol#429)
Variable JetswapPair.price0CumulativeLast (JetswapFactory.sol#274) is too similar to JetswapPair.price1CumulativeLast (JetswapFactory.sol#275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-toosimilar

```
INFO:Detectors:
JetswapFactory.createPair(address,address) (JetswapFactory.sol#470-485) uses literals with too
many digits:
- bytecode = type(address)(JetswapPair).creationCode (JetswapFactory.sol#475)
JetswapFactory.slitherConstructorConstantVariables() (JetswapFactory.sol#451-496) uses literals
with too many digits:
- INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type(address)
(JetswapPair).creationCode)) (JetswapFactory.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:/home/fedrik/reports/JetswapFactory.sol analyzed (11 contracts with 72 detectors), 36
result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
fedrik@fedrik-ThinkPad-T410:~/reports$
fedrik@fedrik-ThinkPad-T410:~/reports$ slither /home/fedrik/reports/JetswapRouter.sol
Compilation warnings/errors on /home/fedrik/reports/JetswapRouter.sol:
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This
contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs"
value!), turning off revert strings, or using libraries.
--> /home/fedrik/reports/JetswapRouter.sol:350:1:
|
350 | contract JetswapRouter is IJetswapRouter02 {
| ^ (Relevant source part starts here and spans across multiple lines).
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This
contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs"
value!), turning off revert strings, or using libraries.
--> JetswapRouter.sol:350:1:
|
350 | contract JetswapRouter is IJetswapRouter02 {
| ^ (Relevant source part starts here and spans across multiple lines).
INFO:Detectors:
JetswapRouter._swap(uint256[],address[],address).i (JetswapRouter.sol#551) is a local variable
never initialized
JetswapRouter._swapSupportingFeeOnTransferTokens(address[],address).i
(JetswapRouter.sol#660) is a local variable never initialized
JetswapLibrary.getAmountsOut(address,uint256,address[]).i (JetswapRouter.sol#310) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
localvariables
INFO:Detectors:
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)
(JetswapRouter.sol#371-398) ignores return value by
IJetswapFactory(factory).createPair(tokenA,tokenB) (JetswapRouter.sol#381)
JetswapRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
(JetswapRouter.sol#441-457) ignores return value by
IJetswapPair(pair).transferFrom(msg.sender,pair,liquidity) (JetswapRouter.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

INFO:Detectors:
JetswapRouter.constructor(address,address)._factory (JetswapRouter.sol#361) lacks a zero-check on :
- factory = _factory (JetswapRouter.sol#362)
JetswapRouter.constructor(address,address)._WETH (JetswapRouter.sol#361) lacks a zero-check on :
- WETH = _WETH (JetswapRouter.sol#363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-addressvalidation
INFO:Detectors:
JetswapRouter._swap(uint256[],address[],address) (JetswapRouter.sol#550-561) has external calls inside a loop:
IJetswapPair(JetswapLibrary.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (JetswapRouter.sol#557-559)
JetswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (JetswapRouter.sol#659-676) has external calls inside a loop: (reserve0,reserve1) = pair.getReserves() (JetswapRouter.sol#667)
JetswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (JetswapRouter.sol#659-676) has external calls inside a loop: amountInput = IERC20(input).balanceOf(address(pair)).sub(reserveInput) (JetswapRouter.sol#669)
JetswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (JetswapRouter.sol#659-676) has external calls inside a loop: pair.swap(amount0Out,amount1Out,to,new bytes(0)) (JetswapRouter.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version=0.6.6 (JetswapRouter.sol#5) allows old versions
solc-0.6.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-ofsolidity
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (JetswapRouter.sol#25-29):
- (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (JetswapRouter.sol#27)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (JetswapRouter.sol#31-35):
- (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (JetswapRouter.sol#33)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (JetswapRouter.sol#37-41):
- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (JetswapRouter.sol#39)
Low level call in TransferHelper.safeTransferETH(address,uint256) (JetswapRouter.sol#43-46):
- (success) = to.call{value: value}(new bytes(0)) (JetswapRouter.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IJetswapRouter01.WETH() (JetswapRouter.sol#51) is not in mixedCase
Function IJetswapPair.DOMAIN_SEPARATOR() (JetswapRouter.sol#199) is not in mixedCase

Function IJetswapPair.PERMIT_TYPEHASH() (JetswapRouter.sol#200) is not in mixedCase
Function IJetswapPair.MINIMUM_LIQUIDITY() (JetswapRouter.sol#217) is not in mixedCase
Variable JetswapRouter.WETH (JetswapRouter.sol#354) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soliditynaming-
conventions
INFO:Detectors:
Variable
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#56) is too similar to
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (JetswapRouter.sol#57)
Variable
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#56) is too similar to
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (JetswapRouter.sol#375)
Variable
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#56) is too similar to
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (JetswapRouter.sol#403)
Variable
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (JetswapRouter.sol#374) is too similar to
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (JetswapRouter.sol#375)
Variable
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (JetswapRouter.sol#374) is too similar to
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (JetswapRouter.sol#403)
Variable
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#402) is too similar to
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (JetswapRouter.sol#403)
Variable
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#402) is too similar to
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (JetswapRouter.sol#57)
Variable
JetswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (JetswapRouter.sol#402) is too similar to
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired

(JetswapRouter.sol#375)
Variable
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired
(JetswapRouter.sol#374) is too similar to
IJetswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).a
mountBDesired (JetswapRouter.sol#57)
Variable
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal
(JetswapRouter.sol#392) is too similar to
JetswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal
(JetswapRouter.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-
toosimilar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
- JetswapRouter.quote(uint256,uint256,uint256) (JetswapRouter.sol#741-743)
getAmountOut(uint256,uint256,uint256) should be declared external:
- JetswapRouter.getAmountOut(uint256,uint256,uint256) (JetswapRouter.sol#745-753)
getAmountIn(uint256,uint256,uint256) should be declared external:
- JetswapRouter.getAmountIn(uint256,uint256,uint256) (JetswapRouter.sol#755-763)
getAmountsOut(uint256,address[]) should be declared external:
- JetswapRouter.getAmountsOut(uint256,address[]) (JetswapRouter.sol#765-773)
getAmountsIn(uint256,address[]) should be declared external:
- JetswapRouter.getAmountsIn(uint256,address[]) (JetswapRouter.sol#775-783)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-
thatcould-
be-declared-external
INFO:Slither:/home/fedrik/reports/JetswapRouter.sol analyzed (10 contracts with 72 detectors), 37
result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
fedrik@fedrik-ThinkPad-T410:~/reports$
fedrik@fedrik-ThinkPad-T410:~/reports$ slither /home/fedrik/reports/WingsToken.sol
Compilation warnings/errors on /home/fedrik/reports/WingsToken.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a
comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use
"SPDXLicense-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more
information.
--> /home/fedrik/reports/WingsToken.sol
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a
comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use
"SPDXLicense-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more
information.
--> WingsToken.sol
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will

need to use the @dev tag explicitly.
--> /home/fedrik/reports/WingsToken.sol:945:5:
|
945 | /// @notice A record of each accounts delegate
| ^^^
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.
--> WingsToken.sol:945:5:
|
945 | /// @notice A record of each accounts delegate
| ^^^
INFO:Detectors:
WingsToken._writeCheckpoint(address,uint32,uint256,uint256) (WingsToken.sol#1152-1178) uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (WingsToken.sol#1165-1166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strictequalities
INFO:Detectors:
BEP20.constructor(string,string).name (WingsToken.sol#630) shadows:
- BEP20.name() (WingsToken.sol#646-648) (function)
- IBEP20.name() (WingsToken.sol#131) (function)
BEP20.constructor(string,string).symbol (WingsToken.sol#630) shadows:
- BEP20.symbol() (WingsToken.sol#660-662) (function)
- IBEP20.symbol() (WingsToken.sol#126) (function)
BEP20.allowance(address,address).owner (WingsToken.sol#698) shadows:
- Ownable.owner() (WingsToken.sol#66-68) (function)
BEP20._approve(address,address,uint256).owner (WingsToken.sol#901) shadows:
- Ownable.owner() (WingsToken.sol#66-68) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
WingsToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (WingsToken.sol#1012-1051) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,CAKE::delegateBySig: signature expired) (WingsToken.sol#1049)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (WingsToken.sol#425-437) uses assembly
- INLINE ASM (WingsToken.sol#433-435)
Address._functionCallWithValue(address,bytes,uint256,string) (WingsToken.sol#552-579) uses assembly
- INLINE ASM (WingsToken.sol#571-574)
WingsToken.getChainId() (WingsToken.sol#1189-1195) uses assembly
- INLINE ASM (WingsToken.sol#1191-1193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (WingsToken.sol#455-467):
- (success) = recipient.call{value: amount}() (WingsToken.sol#462)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string)
(WingsToken.sol#552-579):
- (success,returndata) = target.call{value: weiValue}(data) (WingsToken.sol#561-562)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter WingsToken.mint(address,uint256)._to (WingsToken.sol#934) is not in mixedCase
Parameter WingsToken.mint(address,uint256)._amount (WingsToken.sol#934) is not in mixedCase
Variable WingsToken._delegates (WingsToken.sol#946) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soliditynaming-
conventions
INFO:Detectors:
Redundant expression "this (WingsToken.sol#28)" inContext (WingsToken.sol#18-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (WingsToken.sol#85-88)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (WingsToken.sol#94-96)
decimals() should be declared external:
- BEP20.decimals() (WingsToken.sol#653-655)
symbol() should be declared external:
- BEP20.symbol() (WingsToken.sol#660-662)
totalSupply() should be declared external:
- BEP20.totalSupply() (WingsToken.sol#667-669)
transfer(address,uint256) should be declared external:
- BEP20.transfer(address,uint256) (WingsToken.sol#686-693)
allowance(address,address) should be declared external:
- BEP20.allowance(address,address) (WingsToken.sol#698-705)
approve(address,uint256) should be declared external:
- BEP20.approve(address,uint256) (WingsToken.sol#714-721)
transferFrom(address,address,uint256) should be declared external:
- BEP20.transferFrom(address,address,uint256) (WingsToken.sol#735-750)
increaseAllowance(address,uint256) should be declared external:
- BEP20.increaseAllowance(address,uint256) (WingsToken.sol#764-774)
decreaseAllowance(address,uint256) should be declared external:
- BEP20.decreaseAllowance(address,uint256) (WingsToken.sol#790-803)
mint(uint256) should be declared external:
- BEP20.mint(uint256) (WingsToken.sol#813-816)
mint(address,uint256) should be declared external:
- WingsToken.mint(address,uint256) (WingsToken.sol#934-937)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-thatcould- be-declared-external

Hash0X