QuillAudits

Audit Report
May, 2022

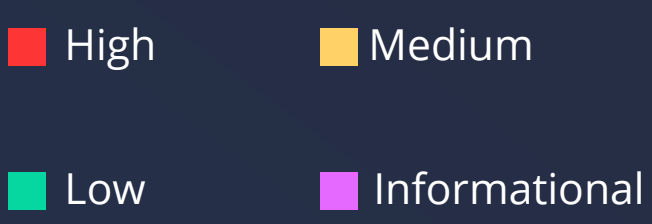
For

ENEDEX

Table of Content

Executive Summary

Project Name	Enedex
Timeline	11 May, 2022 to 24 May, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze and document the Enedex Token and DEX smart contract codebase for quality, security, and correctness.
GitHub Link	*https://github.com/ENEDEX-DEX/SmartContracts*
Commit ID	Version 1 - May 12th c7fb5ac062b20a5075f3cc3db9264e868d9b1f0a Version 2 - May 20th 35e83768a623ddcd6a0b5dbb58dc64f7a2e29447



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	0	0	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract: WNATIVE.sol, WGLMR.sol, WFTM.sol, WBNB.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

B. Contract - UniswapV2Pair.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

C. Contract - UniswapV2Factory.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

D. Contract - UniswapV2Router02.sol

High Severity Issues

1. Wrong init code hash in Library contract

Line #26

```
20  function pairFor(address factory↑, address tokenA↑, address tokenB↑) internal pure returns (address pair↑) {
21      (address token0, address token1) = sortTokens(tokenA↑, tokenB↑);
22      pair↑ = address(uint(keccak256(abi.encodePacked(
23              hex'ff',
24              factory↑,
25              keccak256(abi.encodePacked(token0, token1)),
26              hex'dbe8de032cd5a6eaf8be7cec100683c0cc12320858859782651o2af9a7c6400c' // init code hash
27          ))));
28  }       EneDEX2022, last month • Initial commit …
```

Description

While adding liquidity for the first time, the factory contract will create a Pair contract with some predetermined address. After creation of that new contract, liquidity will be added into that contract by the Router contract which uses the library contract's pairFor function to determine the address of the pair. But the init code hash in the pair contract is incorrect. The liquidity adding functions won't work.

Remediation

It's recommended that you update the UniswapV2Library contract with the correct init code hash.

Status

Fixed

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

E.Contract - Multicall.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

Functional Testing

Some of the tests performed are mentioned below

- WNATIVE tokens should transfer and transferFrom
- WNATIVE tokens should mint on Native token deposit
- WNATIVE tokens should burn tokens on withdraw
- WNATIVE totalSupply should equal the contract balance
- Multicall should aggregate all calls and execute
- Multicall should return the data in array
- Multicall should aggregate all calls and execute
- DEX contracts should allow adding liquidity
- DEX contracts should allow removing liquidity
- DEX contracts should swap correctly
- DEX contracts should support fee on transfer tokens

Automated Testing

```
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (contracts/uniswapv2/UniswapV2Pair.sol#79-92) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (contracts/uniswapv2/UniswapV2Pair.sol#81)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

UniswapV2Router02.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (contracts/uniswapv2/UniswapV2Router02.sol#104-120) ignores return value by IUniswapV2Pair(pair).transferFrom(msg.sender,pair,liquidity) (contracts/uniswapv2/UniswapV2Router02.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

DCTDAO._writeCheckpoint(address,uint32,uint256) (contracts/DCTDAO.sol#107-116) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[account][nCheckpoints - 1].fromBlock == blockNumber (contracts/DCTDAO.sol#110)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

DCTDAOMintable._writeCheckpoint(address,uint32,uint256) (contracts/DCTDAOMintable.sol#133-142) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[account][nCheckpoints - 1].fromBlock == blockNumber (contracts/DCTDAOMintable.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

UniswapV2Pair._safeTransfer(address,address,uint256) (contracts/uniswapv2/UniswapV2Pair.sol#58-53) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),UniswapV2: TRANSFER_FAILED) (contracts/uniswapv2/UniswapV2Pair.sol#52)
UniswapV2Pair.mint(address) (contracts/uniswapv2/UniswapV2Pair.sol#116-144) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/uniswapv2/UniswapV2Pair.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in UniswapV2Pair.burn(address) (contracts/uniswapv2/UniswapV2Pair.sol#147-169):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/uniswapv2/UniswapV2Pair.sol#161)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1) (contracts/uniswapv2/UniswapV2Pair.sol#162)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
                - blockTimestampLast = blockTimestamp (contracts/uniswapv2/UniswapV2Pair.sol#98)
        - kLast = uint256(reserve0).mul(reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#167)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
                - reserve0 = uint112(balance0) (contracts/uniswapv2/UniswapV2Pair.sol#88)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
                - reserve1 = uint112(balance1) (contracts/uniswapv2/UniswapV2Pair.sol#89)
Reentrancy in UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/UniswapV2Factory.sol#30-45):
        External calls:
        - IUniswapV2Pair(pair).initialize(token0,token1) (contracts/uniswapv2/UniswapV2Factory.sol#40)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (contracts/uniswapv2/UniswapV2Factory.sol#41)
        - getPair[token1][token0] = pair (contracts/uniswapv2/UniswapV2Factory.sol#42)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (contracts/uniswapv2/UniswapV2Pair.sol#172-200):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/uniswapv2/UniswapV2Pair.sol#183)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1Out) (contracts/uniswapv2/UniswapV2Pair.sol#184)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (contracts/uniswapv2/UniswapV2Pair.sol#185)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#198)
                - reserve0 = uint112(balance0) (contracts/uniswapv2/UniswapV2Pair.sol#88)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#198)
                - reserve1 = uint112(balance1) (contracts/uniswapv2/UniswapV2Pair.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address).i (contracts/uniswapv2/UniswapV2Router02.sol#323) is a local variable never initialized
UniswapV2Router02._swap(uint256[],address[],address).i (contracts/uniswapv2/UniswapV2Router02.sol#214) is a local variable never initialized
UniswapV2Library.getAmountsOut(address,uint256,address[]).i (contracts/uniswapv2/libraries/UniswapV2Library.sol#68) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256) (contracts/uniswapv2/UniswapV2Router02.sol#94-41) ignores return value by IUniswapV2Factory(factory).createPair(token0,token1) (contracts/uniswapv2/UniswapV2Router02.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

UniswapV2Factory.constructor(address)._feeToSetter (contracts/uniswapv2/UniswapV2Factory.sol#18) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/uniswapv2/UniswapV2Factory.sol#19)
UniswapV2Factory.setFeeTo(address)._feeTo (contracts/uniswapv2/UniswapV2Factory.sol#47) lacks a zero-check on :
                - feeTo = _feeTo (contracts/uniswapv2/UniswapV2Factory.sol#48)
UniswapV2Factory.setMigrator(address)._migrator (contracts/uniswapv2/UniswapV2Factory.sol#52) lacks a zero-check on :
                - migrator = _migrator (contracts/uniswapv2/UniswapV2Factory.sol#54)
UniswapV2Factory.setFeeToSetter(address)._feeToSetter (contracts/uniswapv2/UniswapV2Factory.sol#57) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/uniswapv2/UniswapV2Factory.sol#59)
UniswapV2Pair.initialize(address,address)._token0 (contracts/uniswapv2/UniswapV2Pair.sol#72) lacks a zero-check on :
                - token0 = _token0 (contracts/uniswapv2/UniswapV2Pair.sol#74)
UniswapV2Pair.initialize(address,address)._token1 (contracts/uniswapv2/UniswapV2Pair.sol#72) lacks a zero-check on :
                - token1 = _token1 (contracts/uniswapv2/UniswapV2Pair.sol#75)
UniswapV2Router02.constructor(address,address)._factory (contracts/uniswapv2/UniswapV2Router02.sol#24) lacks a zero-check on :
                - factory = _factory (contracts/uniswapv2/UniswapV2Router02.sol#25)
UniswapV2Router02.constructor(address,address)._WMATIC (contracts/uniswapv2/UniswapV2Router02.sol#24) lacks a zero-check on :
                - WMATIC = _WMATIC (contracts/uniswapv2/UniswapV2Router02.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.aggregate(Multicall.Call[]) (contracts/Multicall.sol#18-26) has external calls inside a loop: (success,ret) = calls[i].target.call{calls[i].callData} (contracts/Multicall.sol#22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in UniswapV2Pair.burn(address) (contracts/uniswapv2/UniswapV2Pair.sol#147-169):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/uniswapv2/UniswapV2Pair.sol#161)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1) (contracts/uniswapv2/UniswapV2Pair.sol#162)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
                - priceOCumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/uniswapv2/UniswapV2Pair.sol#85)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/uniswapv2/UniswapV2Pair.sol#86)
Reentrancy in UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/UniswapV2Factory.sol#30-45):
        External calls:
        - IUniswapV2Pair(pair).initialize(token0,token1) (contracts/uniswapv2/UniswapV2Factory.sol#40)
        State variables written after the call(s):
        - allPairs.push(pair) (contracts/uniswapv2/UniswapV2Factory.sol#43)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (contracts/uniswapv2/UniswapV2Pair.sol#172-200):
```

```
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/uniswapv2/UniswapV2Pair.sol#183)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1Out) (contracts/uniswapv2/UniswapV2Pair.sol#184)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (contracts/uniswapv2/UniswapV2Pair.sol#185)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#198)
                - priceOCumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/uniswapv2/UniswapV2Pair.sol#85)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#198)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/uniswapv2/UniswapV2Pair.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in UniswapV2Pair.burn(address) (contracts/uniswapv2/UniswapV2Pair.sol#147-169):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/uniswapv2/UniswapV2Pair.sol#161)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1) (contracts/uniswapv2/UniswapV2Pair.sol#162)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (contracts/uniswapv2/UniswapV2Pair.sol#168)
        - Sync(reserve0,reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#99)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#166)
Reentrancy in UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/UniswapV2Factory.sol#30-45):
        External calls:
        - IUniswapV2Pair(pair).initialize(token0,token1) (contracts/uniswapv2/UniswapV2Factory.sol#40)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,allPairs.length) (contracts/uniswapv2/UniswapV2Factory.sol#44)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (contracts/uniswapv2/UniswapV2Pair.sol#172-200):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/uniswapv2/UniswapV2Pair.sol#183)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1Out) (contracts/uniswapv2/UniswapV2Pair.sol#184)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (contracts/uniswapv2/UniswapV2Pair.sol#185)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (contracts/uniswapv2/UniswapV2Pair.sol#199)
        - Sync(reserve0,reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#99)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/UniswapV2Pair.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

UniswapV2ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/uniswapv2/UniswapV2ERC20.sol#82-94) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,UniswapV2: EXPIRED) (contracts/uniswapv2/UniswapV2ERC20.sol#83)
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (contracts/uniswapv2/UniswapV2Pair.sol#79-92) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (contracts/uniswapv2/UniswapV2Pair.sol#83)
DeflatingERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/uniswapv2/test/DeflatingERC20.sol#84-96) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,EXPIRED) (contracts/uniswapv2/test/DeflatingERC20.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Address.isContract(address) (contracts/utils/Address.sol#26-39) uses assembly
        - INLINE ASM (contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/utils/Address.sol#147-164) uses assembly
        - INLINE ASM (contracts/utils/Address.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

UniswapV2ERC20.constructor() (contracts/uniswapv2/UniswapV2ERC20.sol#25-30) uses assembly
        - INLINE ASM (contracts/uniswapv2/UniswapV2ERC20.sol#27-29)
UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/UniswapV2Factory.sol#30-45) uses assembly
        - INLINE ASM (contracts/uniswapv2/UniswapV2Factory.sol#37-39)
DeflatingERC20.constructor(uint256) (contracts/uniswapv2/test/DeflatingERC20.sol#13-18) uses assembly
        - INLINE ASM (contracts/uniswapv2/test/DeflatingERC20.sol#25-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['>=0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.12 (contracts/DCTDAOMintable.sol#5)
        - >=0.6.0<0.8.0 (contracts/utils/AccessControl.sol#3)
        - >=0.6.2<0.8.0 (contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (contracts/utils/EnumerableSet.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
        - Version used: ['>=0.6.12', '>=0.5.0', '>=0.5.16', '>=0.6.0', '>=0.6.12', '>=0.6.2']
        - >=0.6.12 (contracts/uniswapv2/UniswapV2ERC20.sol#3)
        - >=0.6.12 (contracts/uniswapv2/UniswapV2Factory.sol#3)
        - >=0.6.12 (contracts/uniswapv2/UniswapV2Pair.sol#3)
        - >=0.6.12 (contracts/uniswapv2/UniswapV2Router02.sol#3)
        - >=0.5.0 (contracts/uniswapv2/interfaces/IERC20.sol#3)
        - >=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Callee.sol#3)
        - >=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Factory.sol#3)
        - >=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#3)
        - >=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#3)
        - >=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router02.sol#3)
        - >=0.5.0 (contracts/uniswapv2/interfaces/IWETH.sol#3)
        - >=0.6.12 (contracts/uniswapv2/libraries/Math.sol#3)
        - >=0.6.12 (contracts/uniswapv2/libraries/SafeMath.sol#3)
        - >=0.6.0 (contracts/uniswapv2/libraries/TransferHelper.sol#3)
        - >=0.6.12 (contracts/uniswapv2/libraries/UQ112x112.sol#4)
        - >=0.5.0 (contracts/uniswapv2/libraries/UniswapV2Library.sol#3)
        - >=0.6.12 (contracts/uniswapv2/test/DeflatingERC20.sol#1)
        - >=0.5.16 (contracts/uniswapv2/test/TestUniswapV2ERC20.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (contracts/utils/AccessControl.sol#285-294) is never used and should be removed
Address._verifyCallResult(bool,bytes,string) (contracts/utils/Address.sol#147-164) is never used and should be removed
Address.functionCall(address,bytes) (contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/utils/Address.sol#114-121) is never used and should be removed
```

```
Address.functionStaticCall(address,bytes,string) (contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (contracts/utils/Address.sol#26-39) is never used and should be removed
Address.sendValue(address,uint256) (contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (contracts/utils/Context.sol#20-23) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (contracts/utils/EnumerableSet.sol#147-149) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (contracts/utils/EnumerableSet.sol#256-258) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (contracts/utils/EnumerableSet.sol#185-187) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (contracts/utils/EnumerableSet.sol#294-296) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (contracts/utils/EnumerableSet.sol#164-166) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (contracts/utils/EnumerableSet.sol#273-275) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (contracts/utils/EnumerableSet.sol#171-173) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (contracts/utils/EnumerableSet.sol#280-282) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (contracts/utils/EnumerableSet.sol#157-159) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (contracts/utils/EnumerableSet.sol#266-268) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

TransferHelper.safeApprove(address,address,uint256) (contracts/uniswapv2/libraries/TransferHelper.sol#7-11) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.6.0<0.8.0 (contracts/utils/AccessControl.sol#3) is too complex
Pragma version=0.6.2<0.8.0 (contracts/utils/Address.sol#3) is too complex
Pragma version=0.6.0<0.8.0 (contracts/utils/Context.sol#3) is too complex
Pragma version=0.6.0<0.8.0 (contracts/utils/EnumerableSet.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/Multicall.sol#5) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IERC20.sol#3) allows old versions
Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Callee.sol#3) allows old versions
Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Factory.sol#3) allows old versions
Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#3) allows old versions
Pragma version=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#3) allows old versions
Pragma version=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router02.sol#3) allows old versions
Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IWETH.sol#3) allows old versions
Pragma version=0.6.0 (contracts/uniswapv2/libraries/TransferHelper.sol#3) allows old versions
Pragma version=0.5.0 (contracts/uniswapv2/libraries/UniswapV2Library.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2ERC20.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/MRAB.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/MFTM.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/MGLMR.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version=0.5.0 (contracts/MNATIVE.sol#1) allows old versions
```

```
Low level call in Address.sendValue(address,uint256) (contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/utils/Address.sol#154-121):
        - (success,returndata) = target.call{value: value}(data) (contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (contracts/utils/Address.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Low level call in Multicall.aggregate(Multicall.Call[]) (contracts/Multicall.sol#18-26):
        - (success,ret) = calls[i].target.call{calls[i].callData} (contracts/Multicall.sol#22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Low level call in UniswapV2Pair._safeTransfer(address,address,uint256) (contracts/uniswapv2/UniswapV2Pair.sol#48-53):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/UniswapV2Pair.sol#51)
Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/uniswapv2/libraries/TransferHelper.sol#7-11):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (contracts/uniswapv2/libraries/TransferHelper.sol#9)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/uniswapv2/libraries/TransferHelper.sol#13-17):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (contracts/uniswapv2/libraries/TransferHelper.sol#15)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (contracts/uniswapv2/libraries/TransferHelper.sol#19-23):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (contracts/uniswapv2/libraries/TransferHelper.sol#21)
Low level call in TransferHelper.safeTransferNATIVE(address,uint256) (contracts/uniswapv2/libraries/TransferHelper.sol#25-28):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/uniswapv2/libraries/TransferHelper.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter DCTDAO.mint(address,uint256)._account (contracts/DCTDAO.sol#29) is not in mixedCase
Parameter DCTDAO.mint(address,uint256)._number (contracts/DCTDAO.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Parameter DCTDAOMintable.mint(address,uint256)._account (contracts/DCTDAOMintable.sol#31) is not in mixedCase
Parameter DCTDAOMintable.mint(address,uint256)._number (contracts/DCTDAOMintable.sol#31) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UniswapV2ERC20.DOMAIN_SEPARATOR (contracts/uniswapv2/UniswapV2ERC20.sol#17) is not in mixedCase
Parameter UniswapV2Factory.setFeeTo(address)._feeTo (contracts/uniswapv2/UniswapV2Factory.sol#47) is not in mixedCase
Parameter UniswapV2Factory.setMigrator(address)._migrator (contracts/uniswapv2/UniswapV2Factory.sol#52) is not in mixedCase
Parameter UniswapV2Factory.setFeeToSetter(address)._feeToSetter (contracts/uniswapv2/UniswapV2Factory.sol#57) is not in mixedCase
Parameter UniswapV2Pair.initialize(address,address)._token0 (contracts/uniswapv2/UniswapV2Pair.sol#72) is not in mixedCase
Parameter UniswapV2Pair.initialize(address,address)._token1 (contracts/uniswapv2/UniswapV2Pair.sol#72) is not in mixedCase
Variable UniswapV2Router02.WNATIVE (contracts/uniswapv2/UniswapV2Router02.sol#17) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#20) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#21) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#38) is not in mixedCase
Function IUniswapV2Router01.WNATIVE() (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#7) is not in mixedCase
Variable DeflatingERC20.DOMAIN_SEPARATOR (contracts/uniswapv2/test/DeflatingERC20.sol#11) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (contracts/uniswapv2/interfaces/IUniswapV2ERC20.sol#20) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (contracts/uniswapv2/interfaces/IUniswapV2ERC20.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/utils/Context.sol#21)" inContext (contracts/utils/Context.sol#15-24)
```

```
Reentrancy in WBNB.withdraw(uint256) (contracts/WBNB.sol#24-29):
        External calls:
        - msg.sender.transfer(wad) (contracts/WBNB.sol#27)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (contracts/WBNB.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Reentrancy in WFTM.withdraw(uint256) (contracts/WFTM.sol#23-29):
        External calls:
        - msg.sender.transfer(wad) (contracts/WFTM.sol#26)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (contracts/WFTM.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Reentrancy in WGLMR.withdraw(uint256) (contracts/WGLMR.sol#23-29):
        External calls:
        - msg.sender.transfer(wad) (contracts/WGLMR.sol#26)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (contracts/WGLMR.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Reentrancy in WNATIVE.withdraw(uint256) (contracts/WNATIVE.sol#23-29):
        External calls:
        - msg.sender.transfer(wad) (contracts/WNATIVE.sol#26)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (contracts/WNATIVE.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable UniswapV2Pair.swap(uint256,uint256,address,bytes).balance0Adjusted (contracts/uniswapv2/UniswapV2Pair.sol#193) is too similar to UniswapV2Pair.swap(uint256,uint256,address,bytes).balance1Adjusted (contracts/uniswapv2/UniswapV2Pair.sol#194)
Variable UniswapV2Pair.price0CumulativeLast (contracts/uniswapv2/UniswapV2Pair.sol#52) is too similar to UniswapV2Pair.price1CumulativeLast (contracts/uniswapv2/UniswapV2Pair.sol#53)
Variable UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#67) is too similar to UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#68)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#52) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#53)
Variable UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#37) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#53)
Variable UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#37) is too similar to UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#38)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#52) is too similar to UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#66)
Variable UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#37) is too similar to UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#66)
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#52) is too similar to UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#38)
Variable UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#65) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#53)
Variable UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/uniswapv2/UniswapV2Router02.sol#65) is too similar to UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/uniswapv2/UniswapV2Router02.sol#66)
Variable UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (contracts/uniswapv2/UniswapV2Router02.sol#55) is too similar to UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (contracts/uniswapv2/UniswapV2Router02.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

OCTDAO.constructor() (contracts/OCTDAO.sol#25-27) uses literals with too many digits:
```

```
        - mint(address(msg.sender),10000000 * 1e18) (contracts/OCTDAO.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

UniswapV2Factory.pairCodeHash() (contracts/uniswapv2/UniswapV2Factory.sol#26-28) uses literals with too many digits:
        - keccak256(bytes)(type()(UniswapV2Pair).creationCode) (contracts/uniswapv2/UniswapV2Factory.sol#27)
UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/UniswapV2Factory.sol#30-45) uses literals with too many digits:
        - bytecode = type()(UniswapV2Pair).creationCode (contracts/uniswapv2/UniswapV2Factory.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

WBNB.decimals (contracts/WBNB.sol#7) should be constant
WBNB.name (contracts/WBNB.sol#5) should be constant
WBNB.symbol (contracts/WBNB.sol#6) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

WFTM.decimals (contracts/WFTM.sol#8) should be constant
WFTM.name (contracts/WFTM.sol#6) should be constant
WFTM.symbol (contracts/WFTM.sol#7) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

WGLMR.decimals (contracts/WGLMR.sol#8) should be constant
WGLMR.name (contracts/WGLMR.sol#6) should be constant
WGLMR.symbol (contracts/WGLMR.sol#7) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

WNATIVE.decimals (contracts/WNATIVE.sol#8) should be constant
WNATIVE.name (contracts/WNATIVE.sol#6) should be constant
WNATIVE.symbol (contracts/WNATIVE.sol#7) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

mint(address,uint256) should be declared external:
        - OCTDAOErc20Enable.mint(address,uint256) (contracts/OCTDAOErc20Enable.sol#31-38)
burnFrom(address,uint256) should be declared external:
        - OCTDAOErc20Enable.burnFrom(address,uint256) (contracts/OCTDAOErc20Enable.sol#51-57)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControl.getRoleMemberCount(bytes32) (contracts/utils/AccessControl.sol#95-97)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControl.getRoleMember(bytes32,uint256) (contracts/utils/AccessControl.sol#111-113)
getRoleAdmin(bytes32) should be declared external:
        - AccessControl.getRoleAdmin(bytes32) (contracts/utils/AccessControl.sol#121-123)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (contracts/utils/AccessControl.sol#131-133)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (contracts/utils/AccessControl.sol#150-154)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (contracts/utils/AccessControl.sol#170-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

aggregate(Multicall.Call[]) should be declared external:
        - Multicall.aggregate(Multicall.Call[]) (contracts/Multicall.sol#18-26)
getEthBalance(address) should be declared external:
        - Multicall.getEthBalance(address) (contracts/Multicall.sol#28-30)
getBlockHash(uint256) should be declared external:
```

```
        - Multicall.getBlockHash(uint256) (contracts/Multicall.sol#31-33)
getLastBlockHash() should be declared external:
        - Multicall.getLastBlockHash() (contracts/Multicall.sol#34-36)
getCurrentBlockTimestamp() should be declared external:
        - Multicall.getCurrentBlockTimestamp() (contracts/Multicall.sol#37-39)
getCurrentBlockDifficulty() should be declared external:
        - Multicall.getCurrentBlockDifficulty() (contracts/Multicall.sol#40-42)
getCurrentBlockGasLimit() should be declared external:
        - Multicall.getCurrentBlockGasLimit() (contracts/Multicall.sol#43-45)
getCurrentBlockCoinbase() should be declared external:
        - Multicall.getCurrentBlockCoinbase() (contracts/Multicall.sol#46-48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

quote(uint256,uint256,uint256) should be declared external:
        - UniswapV2Router02.quote(uint256,uint256,uint256) (contracts/uniswapv2/UniswapV2Router02.sol#404-406)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - UniswapV2Router02.getAmountOut(uint256,uint256,uint256) (contracts/uniswapv2/UniswapV2Router02.sol#408-416)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - UniswapV2Router02.getAmountIn(uint256,uint256,uint256) (contracts/uniswapv2/UniswapV2Router02.sol#418-426)
getAmountsOut(uint256,address[]) should be declared external:
        - UniswapV2Router02.getAmountsOut(uint256,address[]) (contracts/uniswapv2/UniswapV2Router02.sol#428-436)
getAmountsIn(uint256,address[]) should be declared external:
        - UniswapV2Router02.getAmountsIn(uint256,address[]) (contracts/uniswapv2/UniswapV2Router02.sol#438-446)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

withdraw(uint256) should be declared external:
        - WBNB.withdraw(uint256) (contracts/WBNB.sol#24-29)
totalSupply() should be declared external:
        - WBNB.totalSupply() (contracts/WBNB.sol#31-33)
approve(address,uint256) should be declared external:
        - WBNB.approve(address,uint256) (contracts/WBNB.sol#35-39)
transfer(address,uint256) should be declared external:
        - WBNB.transfer(address,uint256) (contracts/WBNB.sol#41-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

withdraw(uint256) should be declared external:
        - WFTM.withdraw(uint256) (contracts/WFTM.sol#23-28)
totalSupply() should be declared external:
        - WFTM.totalSupply() (contracts/WFTM.sol#30-32)
approve(address,uint256) should be declared external:
        - WFTM.approve(address,uint256) (contracts/WFTM.sol#34-38)
transfer(address,uint256) should be declared external:
        - WFTM.transfer(address,uint256) (contracts/WFTM.sol#40-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

withdraw(uint256) should be declared external:
        - WGLMR.withdraw(uint256) (contracts/WGLMR.sol#23-28)
totalSupply() should be declared external:
        - WGLMR.totalSupply() (contracts/WGLMR.sol#30-32)
approve(address,uint256) should be declared external:
        - WGLMR.approve(address,uint256) (contracts/WGLMR.sol#34-38)
transfer(address,uint256) should be declared external:
```

```
        - WGLMR.transfer(address,uint256) (contracts/WGLMR.sol#40-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

withdraw(uint256) should be declared external:
        - WNATIVE.withdraw(uint256) (contracts/WNATIVE.sol#23-28)
totalSupply() should be declared external:
        - WNATIVE.totalSupply() (contracts/WNATIVE.sol#30-32)
approve(address,uint256) should be declared external:
        - WNATIVE.approve(address,uint256) (contracts/WNATIVE.sol#34-38)
transfer(address,uint256) should be declared external:
        - WNATIVE.transfer(address,uint256) (contracts/WNATIVE.sol#40-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (31 contracts with 77 detectors), 175 result(s) found
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. Only one High Severity Issues Found,which the Enedex Team Fixed.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Enedex Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Enedex Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2022

For

ENEDEX

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com