EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	Aceswap Finance
Website:	https://aceswap.finance
Platform:	Polygon (Matic)
Language:	Solidity
Date:	July 9th, 2021

Table of contents

Introduction

EtherAuthority was contracted by the Aceswap team to perform the Security audit of the Aceswap smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on July 9th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Ace allows for swapping of ERC20 compatible tokens across multiple networks.

Audit scope

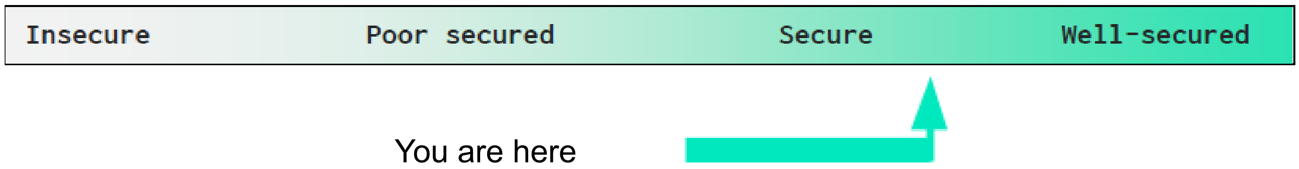
Name	Code Review and Security Analysis Report for Aceswap protocol Smart Contracts
Platform	**Polygon / Solidity**
File 1	AceToken.sol
Smart Contract Online Code	https://polygonscan.com/address/0x550d07A5c1591331598E4e3A38a8C32d41EFc7B7#code
File 1 MD5 Hash	F2328A241CF253808A4991D82C80E9BD
File 2	MasterChef.sol
Smart Contract Online Code	https://polygonscan.com/address/0xEd9a65ED27b69667cDE22f1ac834aE0dB9632C16#code
File 2 MD5 Hash	C37D687746BD08D95FDF133E0D8FC4D9
File 3	UniswapV2Factory.sol
Smart Contract Online Code	https://polygonscan.com/address/0x5013467Ac3A280ede50EF048c13Be05492fDbC3A#code
File 3 MD5 Hash	06385E5CBFF099632D856A242CC02E8E
File 4	UniswapV2Router02.sol
Smart Contract Online Code	https://polygonscan.com/address/0x787c87779E51AfF230954af25733cb27368B19c3#code
File 4 MD5 Hash	7731807FA74843F37AF0773B116B984A
Audit Date	July 9th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1: AceToken.sol ● Name: AceToken ● Symbol: ACE ● Decimals: 18 ● mint: Owner can create a `_amount` token to `_to`.	**YES, This is valid. Owner must be a MasterChef smart contract.**
File 2: MasterChef.sol ● The Masterchef owner can access functions like add and set LP tokens, update ACE per block, Set the migrator contract, etc.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 3: UniswapV2Factory.sol ● The UniswapV2Factory owner can access functions like createpair, setFeeTo, setMigrator, etc.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 4: UniswapV2Router02.sol ● The UniswapV2Router02 can access functions like addLiquidity, remove liquidity, etc.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 4 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 4 smart contracts. Smart contracts also contain Libraries, Smart contract inherits and Interfaces. These are compact and well written contracts.

The libraries in the Aceswap Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Aceswap Protocol.

The Aceswap team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given ACE protocol smart contracts code in the form of a Polygonscan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Aceswap protocol has smart contracts, having functionality like mint, burn, pool, Swap, Liquidity, etc.

AceToken.sol

(1) Interface

(a) IERC20

(2) Inherited contracts

(a) Ownable

(b) ERC20

(3) Struct

(a) Checkpoint

(4) Usages

(a) using SafeMath for uint256;

(5) Events

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(b) event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);

(c) event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	mint	write	Function input parameters lack of check	Refer Audit Findings
2	delegates	external	Passed	No Issue
3	delegate	external	Passed	No Issue

4	delegateBySig	external	Handle signature carefully	No Issue
5	getCurrentVotes	external	Passed	No Issue
6	getPriorVotes	external	Passed	No Issue
7	_delegate	internal	Passed	No Issue
8	_moveDelegates	internal	Passed	No Issue
9	_writeCheckpoint	internal	Passed	No Issue
10	safe32	internal	Passed	No Issue
11	getChainId	internal	Passed	No Issue
12	name	read	Passed	No Issue
13	symbol	read	Passed	No Issue
14	decimals	read	Passed	No Issue
15	totalSupply	read	Passed	No Issue
16	balanceOf	read	Passed	No Issue
17	transfer	write	Passed	No Issue
18	allowance	read	Passed	No Issue
19	approve	write	Passed	No Issue
20	transferFrom	write	Passed	No Issue
21	increaseAllowance	write	Passed	No Issue
22	decreaseAllowance	write	Passed	No Issue
23	_transfer	internal	Passed	No Issue
24	_mint	internal	Passed	No Issue
25	_burn	internal	Passed	No Issue
26	_approve	internal	Passed	No Issue
27	_setupDecimals	internal	Non used functions	Refer Audit Findings
28	_beforeTokenTransfer	internal	Empty function used	Refer Audit Findings
29	owner	read	Passed	No Issue
30	onlyOwner	modifier	Passed	No Issue
31	renounceOwnership	write	access by Owner	No Issue
32	transferOwnership	write	access by Owner	No Issue

MasterChef.sol

(1) Interface

(a) IERC20

(b) IMigratorChef

(2) Inherited contracts

(a) Ownable

(b) ERC20

(c) Context

(d) AceToken

(3) Usages

(a) using SafeMath for uint256;

(b) using SafeERC20 for IERC20;

(4) Struct

(a) PoolInfo

(b) UserInfo

(5) Events

(a) event Deposit(address indexed user, uint256 indexed pid, uint256 amount);

(b) event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);

(c) event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);

(d) event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);

(e) event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	poolLength	external	Passed	No Issue
2	add	write	Input validation missing	Refer Audit Findings
3	set	write	Range validation missing	Refer Audit Findings
4	setAcePerBlock	write	access by Owner	No Issue
5	setMigrator	write	access by Owner	No Issue
6	migrate	write	Missing Events for Significant Transactions	Refer Audit Findings
7	getMultiplier	read	Passed	No Issue
8	pendingAce	external	Passed	No Issue
9	massUpdatePools	write	Infinite loop possibility	Refer Audit Findings

10	updatePool	write	Missing Events for Significant Transactions	Refer Audit Findings
11	deposit	write	Passed	No Issue
12	withdraw	write	Passed	No Issue
13	emergencyWithdraw	write	Values set after transfer tokens	Refer Audit Findings
14	safeAceTransfer	internal	Passed	No Issue
15	dev	write	Passed	No Issue
16	owner	read	Passed	No Issue
17	onlyOwner	modifier	Passed	No Issue
18	renounceOwnership	write	access by Owner	No Issue
19	transferOwnership	write	access by Owner	No Issue
20	_beforeTokenTransfer	write	Empty function used	Refer Audit Findings
21	_setupDecimals	internal	Unused functions	Refer Audit Findings

UniswapV2Factory.sol

(1) Interface

(a) IUniswapV2Factory

(b) IERC20Uniswap

(c) IUniswapV2Callee

(d) IMigrator

(2) Inherited contracts

(a) UniswapV2ERC20

(b) UniswapV2Pair

(3) Usages

(a) address public override feeTo;

(b) address public override feeToSetter;

(c) address public override migrator;

(4) Events

(a) event PairCreated(address indexed token0, address indexed token1, address pair, uint);

(b) event Approval(address indexed owner, address indexed spender, uint value);

(c) event Transfer(address indexed from, address indexed to, uint value);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	allPairsLength	external	Passed	No Issue
2	pairCodeHash	external	Passed	No Issue
3	createPair	external	Passed	No Issue
4	setFeeTo	external	Passed	No Issue
5	setMigrator	external	Passed	No Issue
6	setFeeToSetter	external	Passed	No Issue
7	_mint	internal	Passed	No Issue
8	_burn	internal	Passed	No Issue
9	_approve	write	Passed	No Issue
10	_transfer	write	Passed	No Issue
11	approve	external	Function input parameters lack of check	Refer Audit Findings
12	transfer	external	Function input parameters lack of check	Refer Audit Findings
13	transferFrom	external	Function input parameters lack of check	Refer Audit Findings
14	permit	external	Passed	No Issue

UniswapV2Router02.sol

(1) Interface

(a) IUniswapV2Pair

(b) IUniswapV2Router01

(c) IUniswapV2Router02

(d) IUniswapV2Factory

(e) IERC20Uniswap

(f) IWETH

(2) Inherited contracts

(a) IUniswapV2Router02

(3) Usages

(a) using SafeMathUniswap for uint;

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	ensure	modifier	Passed	No Issue
2	_addLiquidity	internal	Passed	No Issue
3	addLiquidity	external	Passed	No Issue
4	addLiquidityETH	external	Passed	No Issue
5	removeLiquidity	write	Passed	No Issue
6	removeLiquidityETH	write	Passed	No Issue
7	removeLiquidityWithPermit	external	Passed	No Issue
8	removeLiquidityETHWithPermit	external	Passed	No Issue
9	removeLiquidityETHSupportingFeeOnTransferTokens	write	Passed	No Issue
10	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
11	_swap	internal	Passed	No Issue
12	swapExactTokensForTokens	external	Passed	No Issue
13	swapTokensForExactTokens	external	Passed	No Issue
14	swapExactETHForTokens	external	Passed	No Issue
15	swapTokensForExactETH	external	Passed	No Issue
16	swapExactTokensForETH	external	Passed	No Issue
17	swapETHForExactTokens	external	Passed	No Issue
18	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
19	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
20	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
21	swapExactTokensForETHSupportingFeeOnTransferTokens	external	Passed	No Issue
22	quote	write	Passed	No Issue
23	getAmountOut	write	Passed	No Issue
24	getAmountIn	write	Passed	No Issue
25	getAmountsOut	read	Passed	No Issue
26	getAmountsIn	read	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

AceToken.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

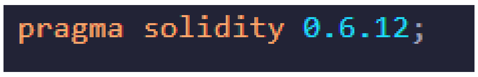
No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

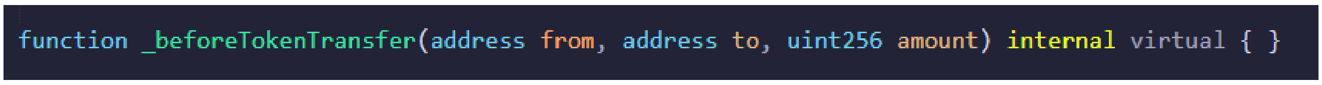


```
pragma solidity 0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Resolution: Please use 0.8.6 which is the latest version.

(2) Empty function used :



```
function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
```

_beforeTokenTransfer this function is empty and used many times but has no effect on code or anything.

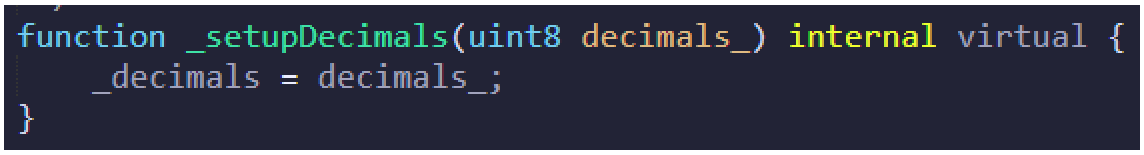
Resolution: Add code in this function or remove this empty function from other functions.

(3) No Max minting of the tokens set :

Setting max minting for the tokens is good for the tokenomics..

Resolution: Set maximum limit of minting tokens, this should be taken care of from the client Side.

(4) Non used functions:

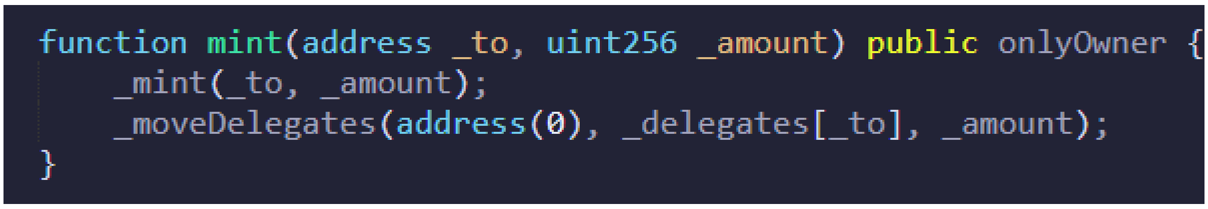


```
function _setupDecimals(uint8 decimals_) internal virtual {
    _decimals = decimals_;
}
```

_setupDecimals function is declared as internal but not used anywhere in the contract.

Resolution: Remove this unused function.

(5) Function input parameters lack of check :

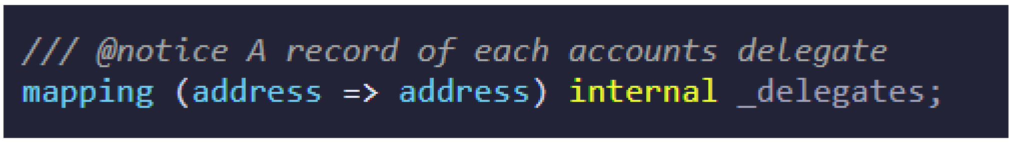


```
function mint(address _to, uint256 _amount) public onlyOwner {
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
}
```

In the mint function there is no more validation amount.

Resolution: It is good practice to check the amount >0 and/or some maximum limit.

(6) Doc tag @notice not valid for statements:



```
/// @notice A record of each accounts delegate
mapping (address => address) internal _delegates;
```

Documentation tags on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.

Resolution: use @dev tag instead of @notice.

UniswapV2Factory.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

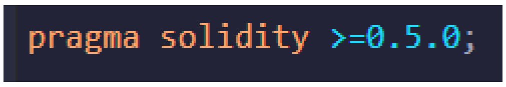
No Medium severity vulnerabilities were found.

Low

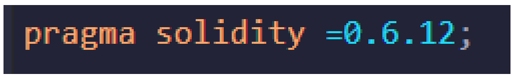
No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:



```
pragma solidity >=0.5.0;
```



```
pragma solidity =0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

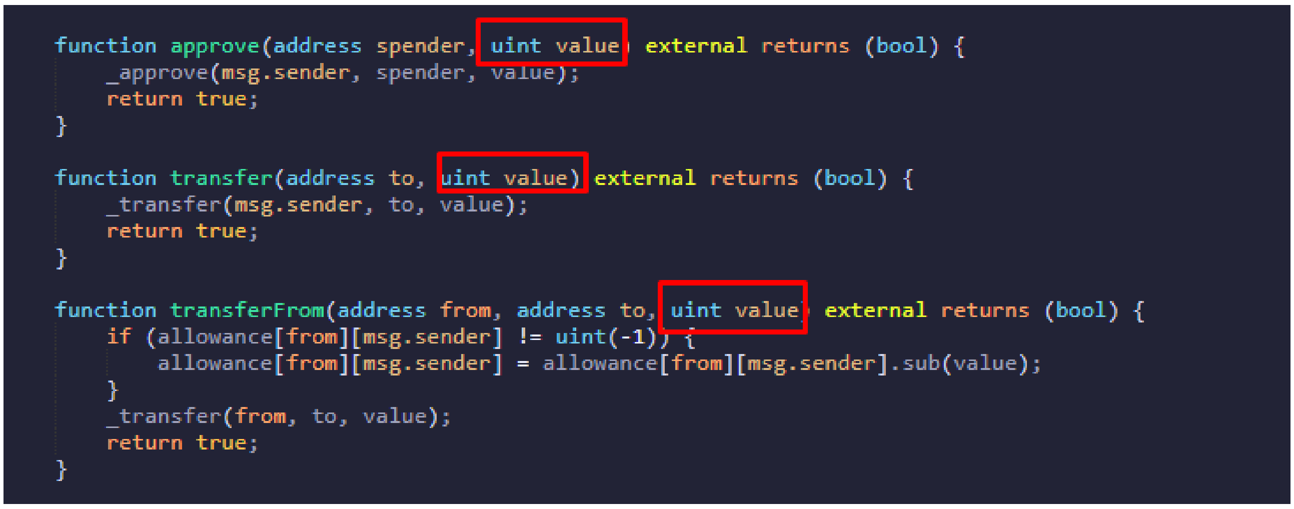
Resolution: Please use 0.8.6 which is the latest version.

(2) No Max minting of the tokens set:

Setting max minting for the tokens is good for tokenomics.

Resolution: Set maximum limit of minting tokens, this should be taken care of from the client Side.

(3) Function input parameters lack of check:



```
function approve(address spender, uint value) external returns (bool) {
    _approve(msg.sender, spender, value);
    return true;
}

function transfer(address to, uint value) external returns (bool) {
    _transfer(msg.sender, to, value);
    return true;
}

function transferFrom(address from, address to, uint value) external returns (bool) {
    if (allowance[from][msg.sender] != uint(-1)) {
        allowance[from][msg.sender] = allowance[from][msg.sender].sub(value);
    }
    _transfer(from, to, value);
    return true;
}
```

In these functions there are no more validation value parameters.

Resolution: It is good practice to check value >0 and/or some maximum limit.

MasterChef.sol

Critical

No critical severity vulnerabilities were found.

High

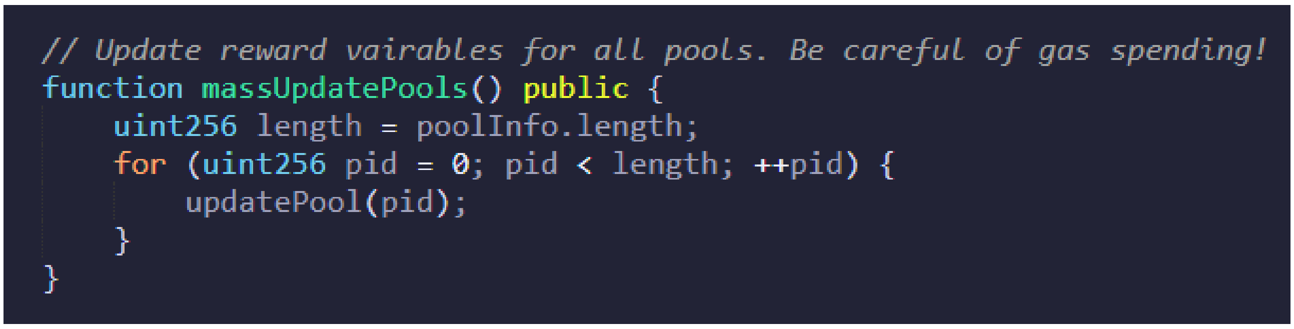
No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loop possibility:



```
// Update reward vairables for all pools. Be careful of gas spending!
function massUpdatePools() public {
    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        updatePool(pid);
    }
}
```

If there are so many pools , then this logic will fail, as it might hit the block's gas limit. If there are very limited pools, then this will work, but will cost more gas.

Resolution: The number of pools should be limited.

(2) Missing Events for Significant Transactions:

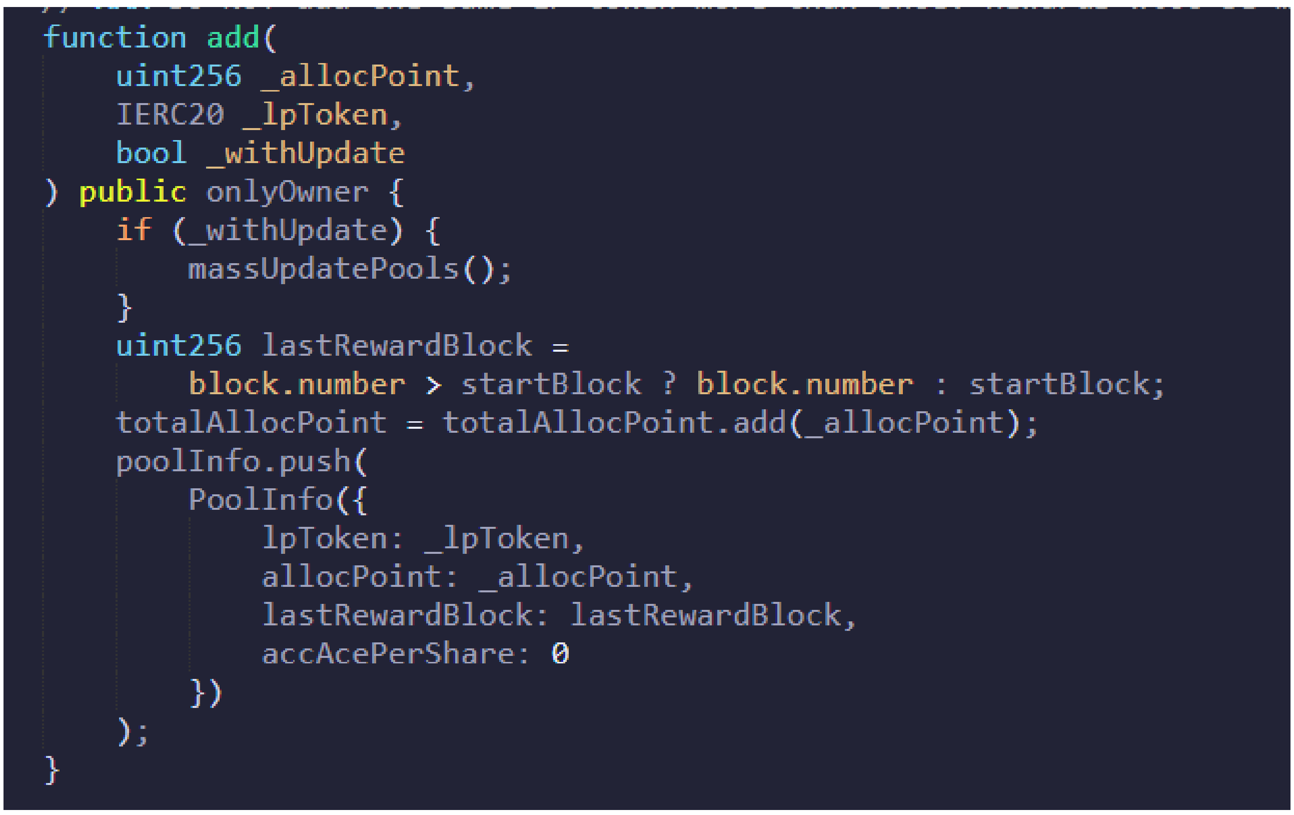
The missing event makes it difficult to track off-chain changes. An event should be emitted for significant transactions calling the following functions:

- add
- set
- migrate
- updatePool

Resolution: We recommend emitting an event to log in following functions:

- add
- set
- migrate
- updatePool

(3) Input validation missing:

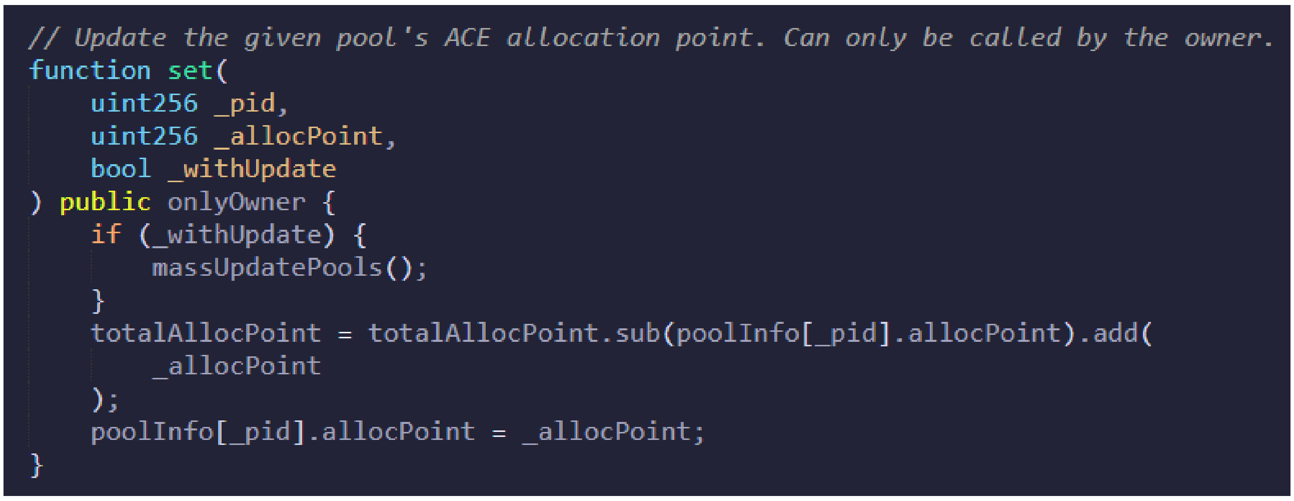


```
function add(
    uint256 _allocPoint,
    IERC20 _lpToken,
    bool _withUpdate
) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    uint256 lastRewardBlock =
        block.number > startBlock ? block.number : startBlock;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolInfo.push(
        PoolInfo({
            lpToken: _lpToken,
            allocPoint: _allocPoint,
            lastRewardBlock: lastRewardBlock,
            accAcePerShare: 0
        })
    );
}
```

As mentioned in the comment, the LP token must not be added twice by human error. It will create a discrepancy in the reward.

Resolution: It is recommended to add an input param check condition to prevent this scenario from happening.

(4) Range validation missing:



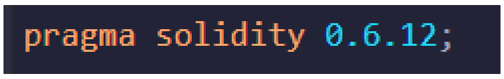
```
// Update the given pool's ACE allocation point. Can only be called by the owner.
function set(
    uint256 _pid,
    uint256 _allocPoint,
    bool _withUpdate
) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(
        _allocPoint
    );
    poolInfo[_pid].allocPoint = _allocPoint;
}
```

The role can set the following state variables arbitrarily large or small causing potential risks in fees and anti whale in set function.

Resolution: We recommend setting ranges and check the following input variables:_pid and _allocPoint in set function.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

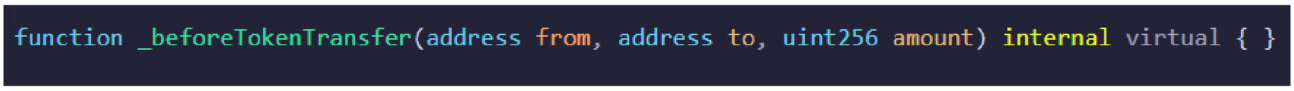


```
pragma solidity 0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Resolution: Please use 0.8.6 which is the latest version.

(2) Empty function used:



```
function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
```

_beforeTokenTransfer this function is empty and used many times but no effect on code or anything

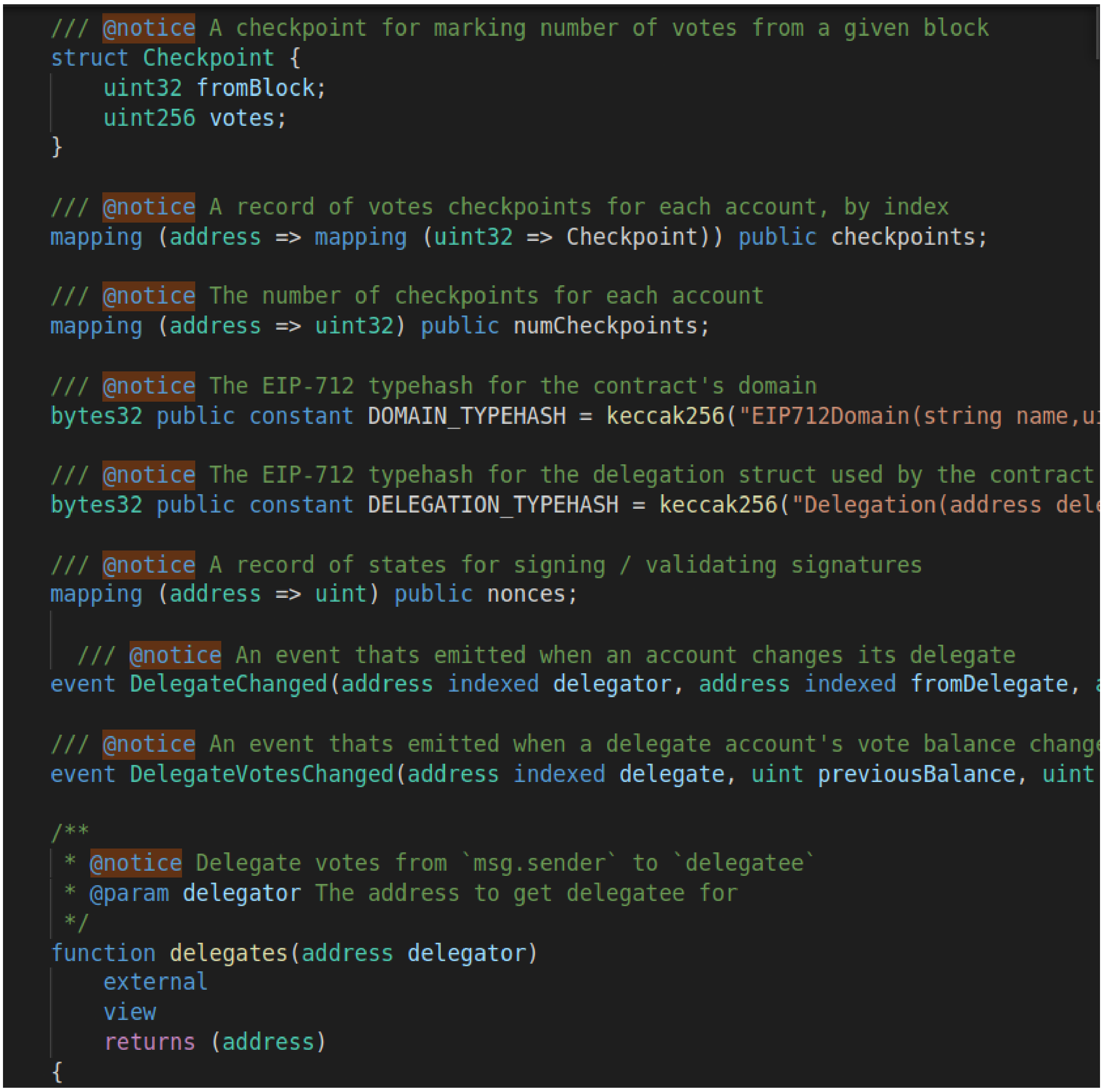
Resolution: Add code in this function or remove this empty function from other functions

(3) No Max minting of the tokens set:

Setting max minting for the tokens is good for tokenomics.

Resolution: Set maximum limit of minting tokens, this should be taken care of from the client Side.

(4) Doc tag @notice not valid for statements:

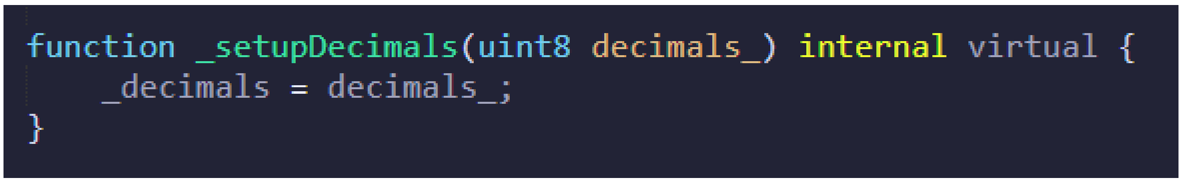


```
/// @notice A checkpoint for marking number of votes from a given block
struct Checkpoint {
    uint32 fromBlock;
    uint256 votes;
}

/// @notice A record of votes checkpoints for each account, by index
mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;

/// @notice The number of checkpoints for each account
mapping (address => uint32) public numCheckpoints;

/// @notice The EIP-712 typehash for the contract's domain
bytes32 public constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(string name,u

/// @notice The EIP-712 typehash for the delegation struct used by the contract
bytes32 public constant DELEGATION_TYPEHASH = keccak256("Delegation(address dele

/// @notice A record of states for signing / validating signatures
mapping (address => uint) public nonces;

  /// @notice An event thats emitted when an account changes its delegate
event DelegateChanged(address indexed delegator, address indexed fromDelegate, a

/// @notice An event thats emitted when a delegate account's vote balance change
event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint

/**
 * @notice Delegate votes from `msg.sender` to `delegatee`
 * @param delegator The address to get delegatee for
 */
function delegates(address delegator)
    external
    view
    returns (address)
{
```

Documentation tags on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.

Resolution: use @dev tag instead of @notice.

(5) Non used functions:

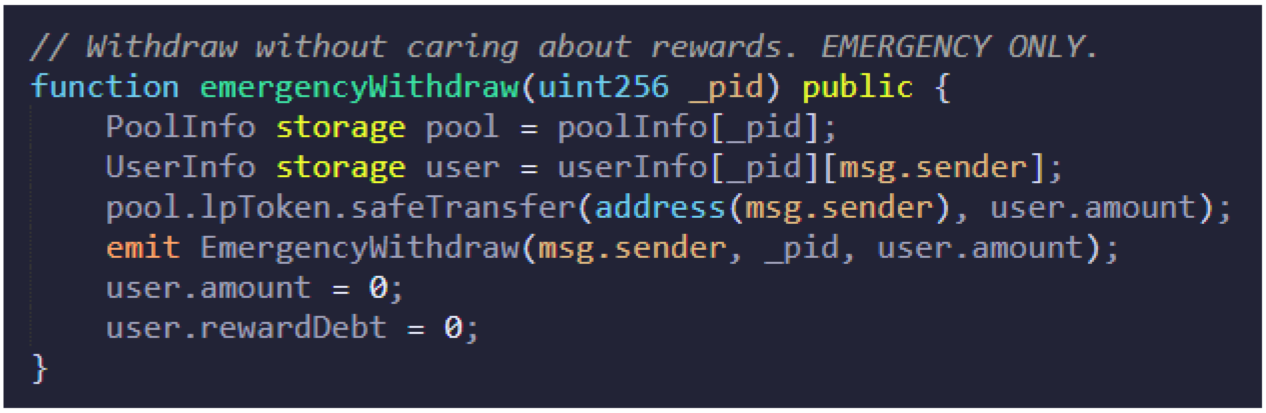


```
function _setupDecimals(uint8 decimals_) internal virtual {
    _decimals = decimals_;
}
```

_setupDecimals function is declared but not used anywhere in the contract.

Resolution: Remove this unused function.

(6) Values set after transfer tokens:



```
// Withdraw without caring about rewards. EMERGENCY ONLY.
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    pool.lpToken.safeTransfer(address(msg.sender), user.amount);
    emit EmergencyWithdraw(msg.sender, _pid, user.amount);
    user.amount = 0;
    user.rewardDebt = 0;
}
```

The emergencyWithdraw function will first withdraw all tokens Of user and then set user.amount=0 and user.rewardDebt=0.

Resolution: We suggest setting user.amount=0 and user.rewardDebt=0 before safeTransfer function to avoid reentrancy.

UniswapV2Router02.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

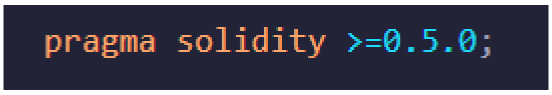
No Medium severity vulnerabilities were found.

Low

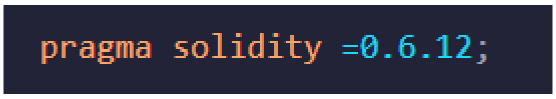
No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:



```
pragma solidity >=0.5.0;
```



```
pragma solidity =0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Resolution: Please use 0.8.6 which is the latest version.

Centralization

These smart contracts have some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- mint: Creates `_amount` token to `_to`. Must be called by the MasterChef contract only.
- Add: Add a new lp to the pool by the MasterChef contract only.
- set: Update the given pool's ACE allocation point by the MasterChef contract only.
- setAcePerBlock: The MasterChef contract can set the given ACE per block.
- setMigrator: The MasterChef contract can set the migrator contract.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview 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 standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

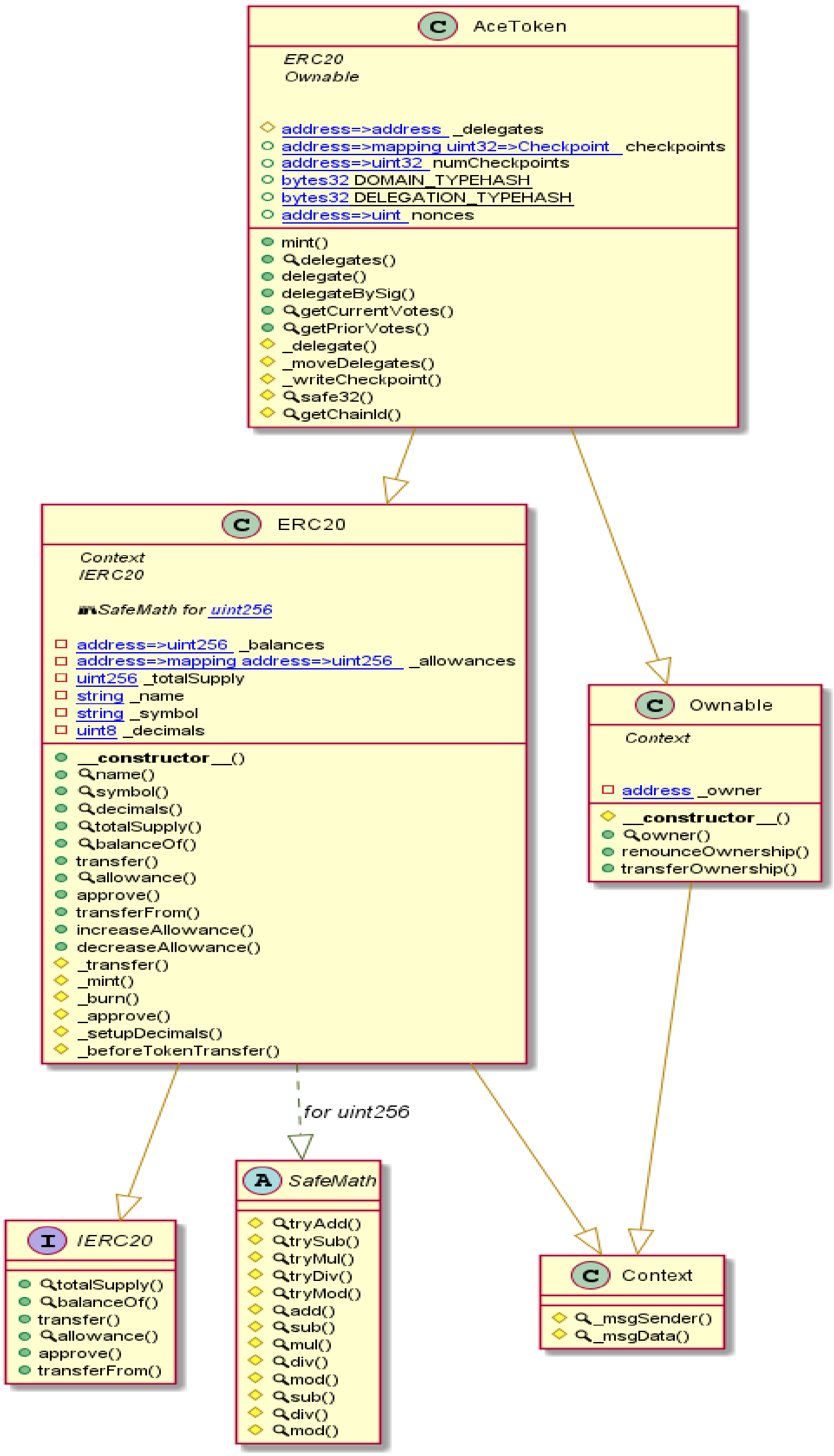
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

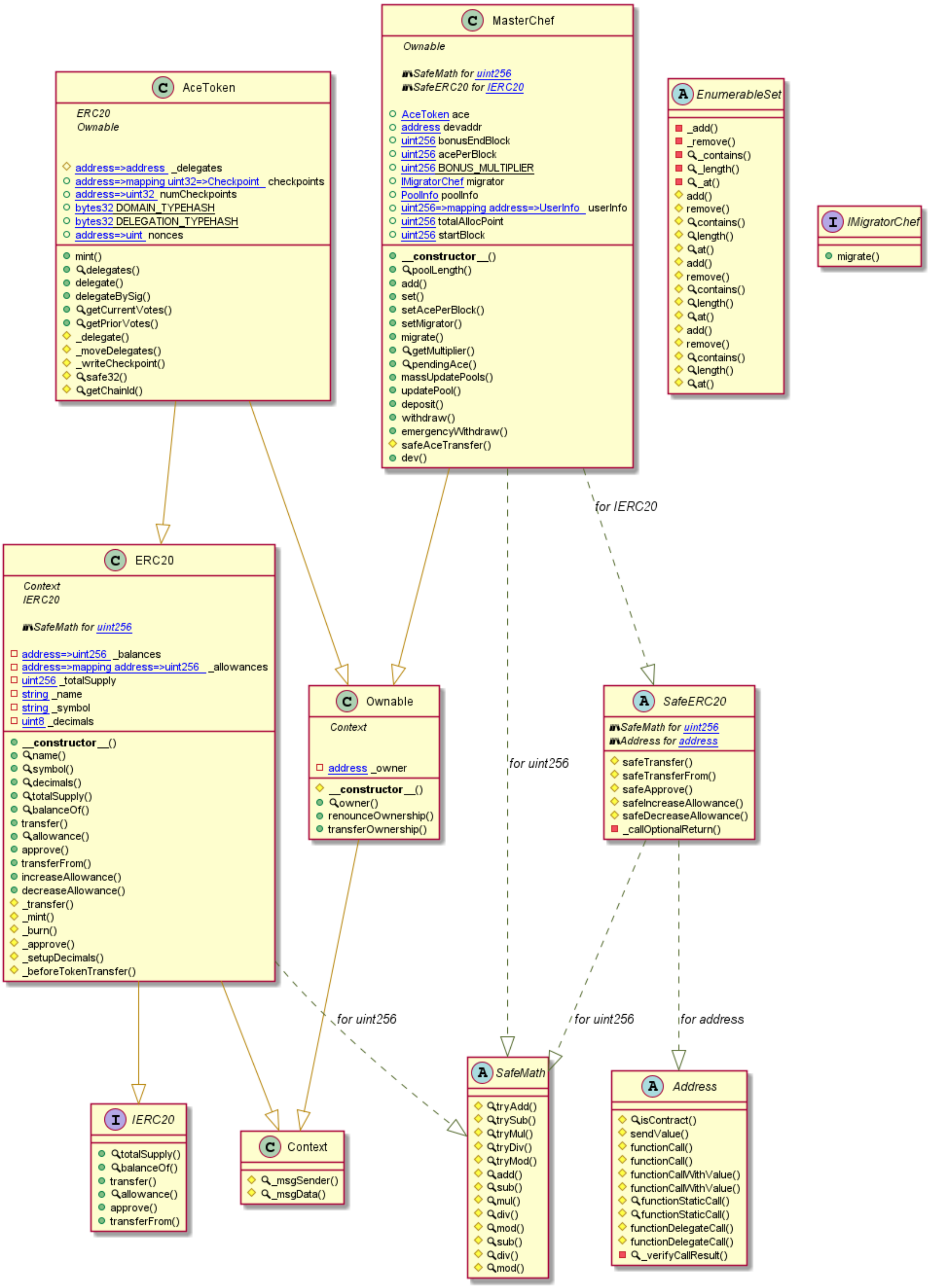
Appendix

Code Flow Diagram - Aceswap Protocol

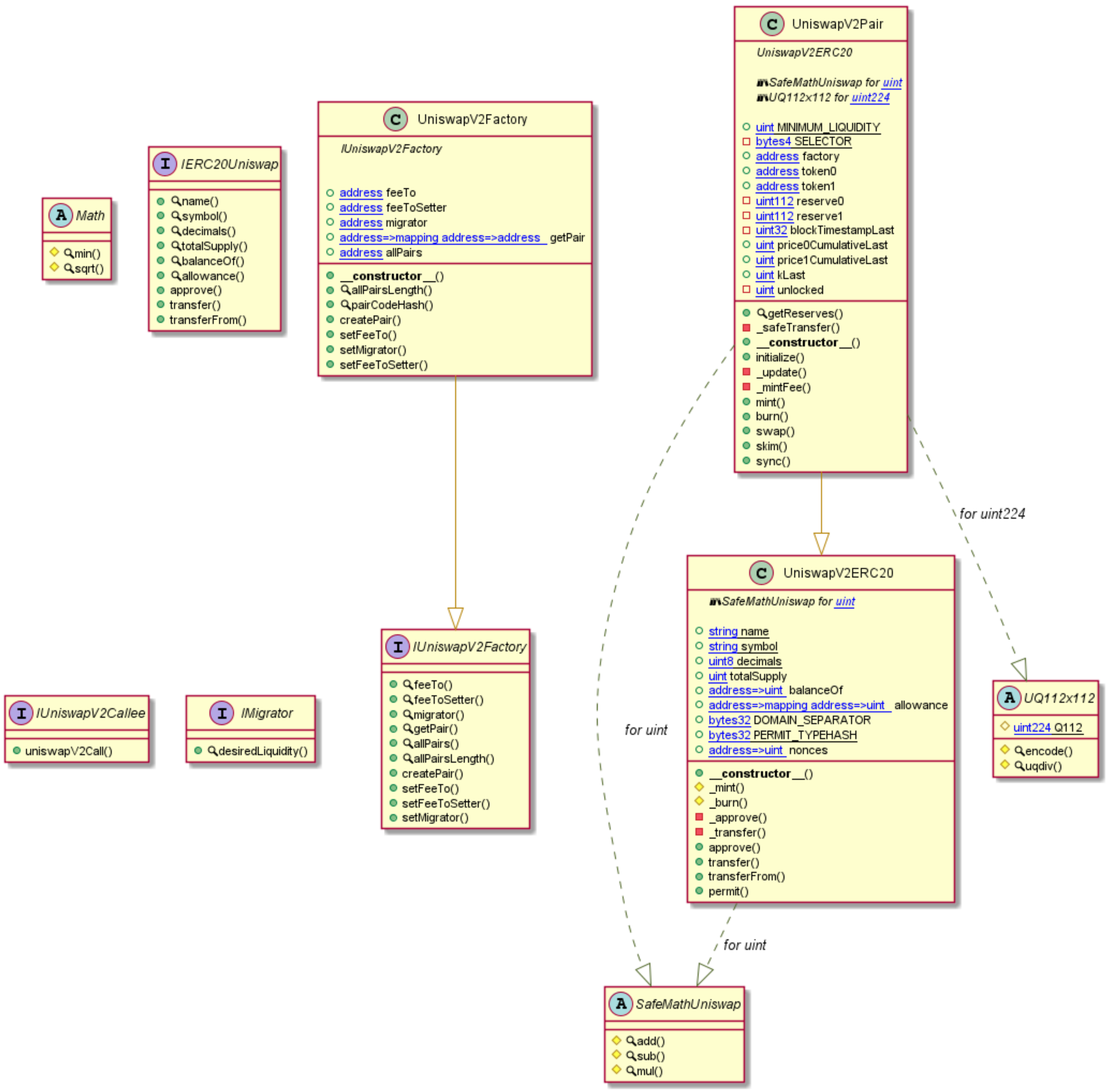
AceToken Diagram



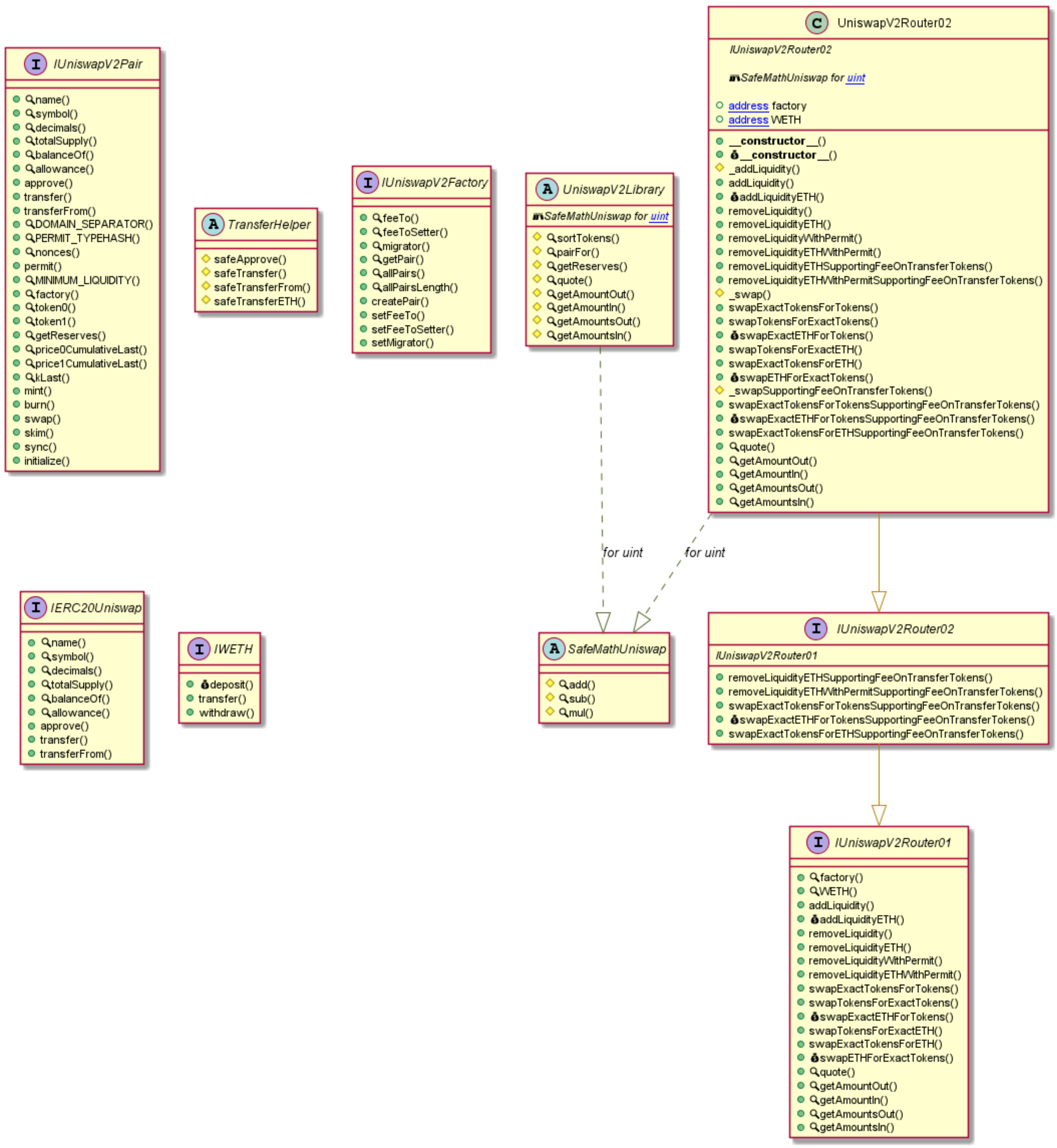
MasterChef Diagram



UniswapV2Factory Diagram



UniswapV2Router02 Diagram



Slither Results Log

Slither log >> AceToken.sol

```
INFO:Detectors:
AceToken._writeCheckpoint(address,uint32,uint256,uint256) (AceToken.sol#912-930) uses a dangerous
strict equality:
	- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber
(AceToken.sol#922)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
AceToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (AceToken.sol#778-819) uses
timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now <= expiry,ACE::delegateBySig: signature expired) (AceToken.sol#817)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AceToken.getChainId() (AceToken.sol#937-941) uses assembly
	- INLINE ASM (AceToken.sol#939)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['0.6.12', '>=0.6.0<0.8.0']
	- >=0.6.0<0.8.0 (AceToken.sol#9)
	- >=0.6.0<0.8.0 (AceToken.sol#34)
	- >=0.6.0<0.8.0 (AceToken.sol#112)
	- >=0.6.0<0.8.0 (AceToken.sol#327)
	- >=0.6.0<0.8.0 (AceToken.sol#630)
	- 0.6.12 (AceToken.sol#698)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (AceToken.sol#26-29) is never used and should be removed
ERC20._burn(address,uint256) (AceToken.sol#569-577) is never used and should be removed
ERC20._setupDecimals(uint8) (AceToken.sol#607-609) is never used and should be removed
SafeMath.div(uint256,uint256) (AceToken.sol#244-247) is never used and should be removed
SafeMath.div(uint256,uint256,string) (AceToken.sol#299-302) is never used and should be removed
SafeMath.mod(uint256,uint256) (AceToken.sol#261-264) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (AceToken.sol#319-322) is never used and should be removed
SafeMath.mul(uint256,uint256) (AceToken.sol#225-230) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (AceToken.sol#133-137) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (AceToken.sol#169-172) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (AceToken.sol#179-182) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (AceToken.sol#154-162) is never used and should be removed
SafeMath.trySub(uint256,uint256) (AceToken.sol#144-147) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (AceToken.sol#9) is too complex
Pragma version>=0.6.0<0.8.0 (AceToken.sol#34) is too complex
Pragma version>=0.6.0<0.8.0 (AceToken.sol#112) is too complex
Pragma version>=0.6.0<0.8.0 (AceToken.sol#327) is too complex
Pragma version>=0.6.0<0.8.0 (AceToken.sol#630) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter AceToken.mint(address,uint256)._to (AceToken.sol#708) is not in mixedCase
Parameter AceToken.mint(address,uint256)._amount (AceToken.sol#708) is not in mixedCase
Variable AceToken._delegates (AceToken.sol#720) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (AceToken.sol#27)" inContext (AceToken.sol#21-30)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
symbol() should be declared external:
	- ERC20.symbol() (AceToken.sol#392-394)
decimals() should be declared external:
	- ERC20.decimals() (AceToken.sol#409-411)
totalSupply() should be declared external:
	- ERC20.totalSupply() (AceToken.sol#416-418)
transfer(address,uint256) should be declared external:
	- ERC20.transfer(address,uint256) (AceToken.sol#435-438)
allowance(address,address) should be declared external:
	- ERC20.allowance(address,address) (AceToken.sol#443-445)
approve(address,uint256) should be declared external:
	- ERC20.approve(address,uint256) (AceToken.sol#454-457)
transferFrom(address,address,uint256) should be declared external:
	- ERC20.transferFrom(address,address,uint256) (AceToken.sol#472-476)
increaseAllowance(address,uint256) should be declared external:
	- ERC20.increaseAllowance(address,uint256) (AceToken.sol#490-493)
decreaseAllowance(address,uint256) should be declared external:
	- ERC20.decreaseAllowance(address,uint256) (AceToken.sol#509-512)
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (AceToken.sol#680-683)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (AceToken.sol#689-693)
mint(address,uint256) should be declared external:
	- AceToken.mint(address,uint256) (AceToken.sol#708-711)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AceToken.sol analyzed (6 contracts with 75 detectors), 38 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MasterChef.sol

```
INFO:Detectors:
MasterChef.safeAceTransfer(address,uint256) (MasterChef.sol#1786-1793) ignores return value by
ace.transfer(_to,aceBal) (MasterChef.sol#1789)
MasterChef.safeAceTransfer(address,uint256) (MasterChef.sol#1786-1793) ignores return value by
ace.transfer(_to,_amount) (MasterChef.sol#1791)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
MasterChef.pendingAce(uint256,address) (MasterChef.sol#1681-1702) performs a multiplication on the
result of a division:
	-aceReward = multiplier.mul(acePerBlock).mul(pool.allocPoint).div(totalAllocPoint)
(MasterChef.sol#1693-1696)
	-accAcePerShare = accAcePerShare.add(aceReward.mul(1e12).div(lpSupply))
(MasterChef.sol#1697-1699)
MasterChef.updatePool(uint256) (MasterChef.sol#1713-1734) performs a multiplication on the result of a
division:
	-aceReward = multiplier.mul(acePerBlock).mul(pool.allocPoint).div(totalAllocPoint)
(MasterChef.sol#1724-1727)
	-pool.accAcePerShare = pool.accAcePerShare.add(aceReward.mul(1e12).div(lpSupply))
(MasterChef.sol#1730-1732)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
AceToken._writeCheckpoint(address,uint32,uint256,uint256) (MasterChef.sol#1472-1490) uses a dangerous
strict equality:
	- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber
(MasterChef.sol#1482)
MasterChef.migrate(uint256) (MasterChef.sol#1651-1660) uses a dangerous strict equality:
	- require(bool,string)(bal == newLpToken.balanceOf(address(this)),migrate: bad) (MasterChef.sol#1658)
MasterChef.updatePool(uint256) (MasterChef.sol#1713-1734) uses a dangerous strict equality:
	- lpSupply == 0 (MasterChef.sol#1719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#1602-1621):
	External calls:
	- massUpdatePools() (MasterChef.sol#1608)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	State variables written after the call(s):
	- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0)) (MasterChef.sol#1613-1620)
	- totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterChef.sol#1612)
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1737-1756):
	External calls:
	- updatePool(_pid) (MasterChef.sol#1740)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	- safeAceTransfer(msg.sender,pending) (MasterChef.sol#1746)
		- ace.transfer(_to,aceBal) (MasterChef.sol#1789)
		- ace.transfer(_to,_amount) (MasterChef.sol#1791)
	- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(MasterChef.sol#1748-1752)
	State variables written after the call(s):
	- user.amount = user.amount.add(_amount) (MasterChef.sol#1753)
	- user.rewardDebt = user.amount.mul(pool.accAcePerShare).div(1e12) (MasterChef.sol#1754)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1776-1783):
	External calls:
	- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1779)
	State variables written after the call(s):
	- user.amount = 0 (MasterChef.sol#1781)
	- user.rewardDebt = 0 (MasterChef.sol#1782)
Reentrancy in MasterChef.migrate(uint256) (MasterChef.sol#1651-1660):
	External calls:
	- lpToken.safeApprove(address(migrator),bal) (MasterChef.sol#1656)
	- newLpToken = migrator.migrate(lpToken) (MasterChef.sol#1657)
	State variables written after the call(s):
	- pool.lpToken = newLpToken (MasterChef.sol#1659)
Reentrancy in MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1624-1636):
	External calls:
	- massUpdatePools() (MasterChef.sol#1630)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	State variables written after the call(s):
	- poolInfo[_pid].allocPoint = _allocPoint (MasterChef.sol#1635)
	- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
(MasterChef.sol#1632-1634)
Reentrancy in MasterChef.updatePool(uint256) (MasterChef.sol#1713-1734):
	External calls:
	- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
	- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	State variables written after the call(s):
	- pool.accAcePerShare = pool.accAcePerShare.add(aceReward.mul(1e12).div(lpSupply))
(MasterChef.sol#1730-1732)
	- pool.lastRewardBlock = block.number (MasterChef.sol#1733)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1759-1773):
	External calls:
	- updatePool(_pid) (MasterChef.sol#1763)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	- safeAceTransfer(msg.sender,pending) (MasterChef.sol#1768)
		- ace.transfer(_to,aceBal) (MasterChef.sol#1789)
		- ace.transfer(_to,_amount) (MasterChef.sol#1791)
	State variables written after the call(s):
	- user.amount = user.amount.sub(_amount) (MasterChef.sol#1769)
	- user.rewardDebt = user.amount.mul(pool.accAcePerShare).div(1e12) (MasterChef.sol#1770)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
MasterChef.constructor(AceToken,address,uint256,uint256,uint256)._devaddr (MasterChef.sol#1584) lacks
a zero-check on :
		- devaddr = _devaddr (MasterChef.sol#1590)
MasterChef.dev(address)._devaddr (MasterChef.sol#1796) lacks a zero-check on :
		- devaddr = _devaddr (MasterChef.sol#1798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1737-1756):
	External calls:
	- updatePool(_pid) (MasterChef.sol#1740)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	- safeAceTransfer(msg.sender,pending) (MasterChef.sol#1746)
		- ace.transfer(_to,aceBal) (MasterChef.sol#1789)
		- ace.transfer(_to,_amount) (MasterChef.sol#1791)
	- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(MasterChef.sol#1748-1752)
	Event emitted after the call(s):
	- Deposit(msg.sender,_pid,_amount) (MasterChef.sol#1755)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1776-1783):
	External calls:
	- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1779)
	Event emitted after the call(s):
	- EmergencyWithdraw(msg.sender,_pid,user.amount) (MasterChef.sol#1780)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1759-1773):
	External calls:
	- updatePool(_pid) (MasterChef.sol#1763)
		- ace.mint(devaddr,aceReward.div(10)) (MasterChef.sol#1728)
		- ace.mint(address(this),aceReward) (MasterChef.sol#1729)
	- safeAceTransfer(msg.sender,pending) (MasterChef.sol#1768)
		- ace.transfer(_to,aceBal) (MasterChef.sol#1789)
		- ace.transfer(_to,_amount) (MasterChef.sol#1791)
	- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1771)
	Event emitted after the call(s):
	- Withdraw(msg.sender,_pid,_amount) (MasterChef.sol#1772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AceToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1338-1379) uses
timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now <= expiry,ACE::delegateBySig: signature expired) (MasterChef.sol#1377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MasterChef.sol#325-334) uses assembly
	- INLINE ASM (MasterChef.sol#332)
Address._verifyCallResult(bool,bytes,string) (MasterChef.sol#470-487) uses assembly
	- INLINE ASM (MasterChef.sol#479-482)
AceToken.getChainId() (MasterChef.sol#1497-1501) uses assembly
	- INLINE ASM (MasterChef.sol#1499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
	- >=0.6.0<0.8.0 (MasterChef.sol#9)
	- >=0.6.0<0.8.0 (MasterChef.sol#87)
	- >=0.6.2<0.8.0 (MasterChef.sol#302)
	- >=0.6.0<0.8.0 (MasterChef.sol#492)
	- >=0.6.0<0.8.0 (MasterChef.sol#564)
	- >=0.6.0<0.8.0 (MasterChef.sol#862)
	- >=0.6.0<0.8.0 (MasterChef.sol#887)
	- >=0.6.0<0.8.0 (MasterChef.sol#955)
	- 0.6.12 (MasterChef.sol#1258)
	- 0.6.12 (MasterChef.sol#1506)
```

```
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (MasterChef.sol#378-380) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterChef.sol#403-405) is never used and should
be removed
Address.functionDelegateCall(address,bytes) (MasterChef.sol#452-454) is never used and should be
removed
Address.functionDelegateCall(address,bytes,string) (MasterChef.sol#462-468) is never used and should be
removed
Address.functionStaticCall(address,bytes) (MasterChef.sol#428-430) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (MasterChef.sol#438-444) is never used and should be
removed
Address.sendValue(address,uint256) (MasterChef.sol#352-358) is never used and should be removed
Context._msgData() (MasterChef.sol#879-882) is never used and should be removed
ERC20._burn(address,uint256) (MasterChef.sol#1197-1205) is never used and should be removed
ERC20._setupDecimals(uint8) (MasterChef.sol#1235-1237) is never used and should be removed
EnumerableSet._add(EnumerableSet.Set,bytes32) (MasterChef.sol#615-625) is never used and should be
removed
EnumerableSet._at(EnumerableSet.Set,uint256) (MasterChef.sol#691-694) is never used and should be
removed
EnumerableSet._contains(EnumerableSet.Set,bytes32) (MasterChef.sol#670-672) is never used and should
be removed
EnumerableSet._length(EnumerableSet.Set) (MasterChef.sol#677-679) is never used and should be
removed
EnumerableSet._remove(EnumerableSet.Set,bytes32) (MasterChef.sol#633-665) is never used and should
be removed
EnumerableSet.add(EnumerableSet.AddressSet,address) (MasterChef.sol#762-764) is never used and
should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (MasterChef.sol#708-710) is never used and
should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (MasterChef.sol#817-819) is never used and should be
removed
EnumerableSet.at(EnumerableSet.AddressSet,uint256) (MasterChef.sol#800-802) is never used and should
be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (MasterChef.sol#746-748) is never used and should
be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (MasterChef.sol#855-857) is never used and should be
removed
EnumerableSet.contains(EnumerableSet.AddressSet,address) (MasterChef.sol#779-781) is never used and
should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (MasterChef.sol#725-727) is never used and
should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (MasterChef.sol#834-836) is never used and
should be removed
EnumerableSet.length(EnumerableSet.AddressSet) (MasterChef.sol#786-788) is never used and should be
removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (MasterChef.sol#732-734) is never used and should be
removed
EnumerableSet.length(EnumerableSet.UintSet) (MasterChef.sol#841-843) is never used and should be
removed
EnumerableSet.remove(EnumerableSet.AddressSet,address) (MasterChef.sol#772-774) is never used and
should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (MasterChef.sol#718-720) is never used and
should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (MasterChef.sol#827-829) is never used and
should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MasterChef.sol#538-541) is never used and
should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MasterChef.sol#533-536) is never used and
should be removed
SafeMath.div(uint256,uint256,string) (MasterChef.sol#274-277) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterChef.sol#236-239) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterChef.sol#294-297) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (MasterChef.sol#108-112) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (MasterChef.sol#144-147) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (MasterChef.sol#154-157) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (MasterChef.sol#129-137) is never used and should be removed
SafeMath.trySub(uint256,uint256) (MasterChef.sol#119-122) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#9) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#87) is too complex
Pragma version>=0.6.2<0.8.0 (MasterChef.sol#302) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#492) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#564) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#862) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#887) is too complex
Pragma version>=0.6.0<0.8.0 (MasterChef.sol#955) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterChef.sol#352-358):
	- (success) = recipient.call{value: amount}() (MasterChef.sol#356)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#413-420):
	- (success,returndata) = target.call{value: value}(data) (MasterChef.sol#418)
Low level call in Address.functionStaticCall(address,bytes,string) (MasterChef.sol#438-444):
	- (success,returndata) = target.staticcall(data) (MasterChef.sol#442)
Low level call in Address.functionDelegateCall(address,bytes,string) (MasterChef.sol#462-468):
	- (success,returndata) = target.delegatecall(data) (MasterChef.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter AceToken.mint(address,uint256)._to (MasterChef.sol#1268) is not in mixedCase
Parameter AceToken.mint(address,uint256)._amount (MasterChef.sol#1268) is not in mixedCase
Variable AceToken._delegates (MasterChef.sol#1280) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,bool)._allocPoint (MasterChef.sol#1603) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,bool)._lpToken (MasterChef.sol#1604) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,bool)._withUpdate (MasterChef.sol#1605) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._pid (MasterChef.sol#1625) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._allocPoint (MasterChef.sol#1626) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._withUpdate (MasterChef.sol#1627) is not in mixedCase
Parameter MasterChef.setMigrator(IMigratorChef)._migrator (MasterChef.sol#1646) is not in mixedCase
Parameter MasterChef.migrate(uint256)._pid (MasterChef.sol#1651) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._from (MasterChef.sol#1663) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (MasterChef.sol#1663) is not in mixedCase
Parameter MasterChef.pendingAce(uint256,address)._pid (MasterChef.sol#1681) is not in mixedCase
Parameter MasterChef.pendingAce(uint256,address)._user (MasterChef.sol#1681) is not in mixedCase
Parameter MasterChef.updatePool(uint256)._pid (MasterChef.sol#1713) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._pid (MasterChef.sol#1737) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._amount (MasterChef.sol#1737) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (MasterChef.sol#1759) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (MasterChef.sol#1759) is not in mixedCase
Parameter MasterChef.emergencyWithdraw(uint256)._pid (MasterChef.sol#1776) is not in mixedCase
Parameter MasterChef.safeAceTransfer(address,uint256)._to (MasterChef.sol#1786) is not in mixedCase
Parameter MasterChef.safeAceTransfer(address,uint256)._amount (MasterChef.sol#1786) is not in
mixedCase
Parameter MasterChef.dev(address)._devaddr (MasterChef.sol#1796) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MasterChef.sol#880)" inContext (MasterChef.sol#874-883)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (MasterChef.sol#937-940)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (MasterChef.sol#946-950)
symbol() should be declared external:
	- ERC20.symbol() (MasterChef.sol#1020-1022)
decimals() should be declared external:
	- ERC20.decimals() (MasterChef.sol#1037-1039)
```

```
totalSupply() should be declared external:
	- ERC20.totalSupply() (MasterChef.sol#1044-1046)
transfer(address,uint256) should be declared external:
	- ERC20.transfer(address,uint256) (MasterChef.sol#1063-1066)
allowance(address,address) should be declared external:
	- ERC20.allowance(address,address) (MasterChef.sol#1071-1073)
approve(address,uint256) should be declared external:
	- ERC20.approve(address,uint256) (MasterChef.sol#1082-1085)
transferFrom(address,address,uint256) should be declared external:
	- ERC20.transferFrom(address,address,uint256) (MasterChef.sol#1100-1104)
increaseAllowance(address,uint256) should be declared external:
	- ERC20.increaseAllowance(address,uint256) (MasterChef.sol#1118-1121)
decreaseAllowance(address,uint256) should be declared external:
	- ERC20.decreaseAllowance(address,uint256) (MasterChef.sol#1137-1140)
mint(address,uint256) should be declared external:
	- AceToken.mint(address,uint256) (MasterChef.sol#1268-1271)
add(uint256,IERC20,bool) should be declared external:
	- MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#1602-1621)
set(uint256,uint256,bool) should be declared external:
	- MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1624-1636)
setAcePerBlock(uint256) should be declared external:
	- MasterChef.setAcePerBlock(uint256) (MasterChef.sol#1639-1643)
setMigrator(IMigratorChef) should be declared external:
	- MasterChef.setMigrator(IMigratorChef) (MasterChef.sol#1646-1648)
migrate(uint256) should be declared external:
	- MasterChef.migrate(uint256) (MasterChef.sol#1651-1660)
deposit(uint256,uint256) should be declared external:
	- MasterChef.deposit(uint256,uint256) (MasterChef.sol#1737-1756)
withdraw(uint256,uint256) should be declared external:
	- MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1759-1773)
emergencyWithdraw(uint256) should be declared external:
	- MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1776-1783)
dev(address) should be declared external:
	- MasterChef.dev(address) (MasterChef.sol#1796-1799)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (11 contracts with 75 detectors), 122 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> UniswapV2Factory.sol

```
INFO:Detectors:
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (UniswapV2Factory.sol#291-304) uses a weak
PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (UniswapV2Factory.sol#293)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
UniswapV2Pair._safeTransfer(address,address,uint256) (UniswapV2Factory.sol#262-265) uses a dangerous
strict equality:
	- require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),UniswapV2:
TRANSFER_FAILED) (UniswapV2Factory.sol#264)
UniswapV2Pair.mint(address) (UniswapV2Factory.sol#328-356) uses a dangerous strict equality:
	- _totalSupply == 0 (UniswapV2Factory.sol#337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in UniswapV2Pair.burn(address) (UniswapV2Factory.sol#359-381):
	External calls:
	- _safeTransfer(_token0,to,amount0) (UniswapV2Factory.sol#373)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1) (UniswapV2Factory.sol#374)
```

```
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	State variables written after the call(s):
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
		- blockTimestampLast = blockTimestamp (UniswapV2Factory.sol#302)
	- kLast = uint256(reserve0).mul(reserve1) (UniswapV2Factory.sol#379)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
		- reserve0 = uint112(balance0) (UniswapV2Factory.sol#300)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
		- reserve1 = uint112(balance1) (UniswapV2Factory.sol#301)
Reentrancy in UniswapV2Factory.createPair(address,address) (UniswapV2Factory.sol#454-469):
	External calls:
	- UniswapV2Pair(pair).initialize(token0,token1) (UniswapV2Factory.sol#464)
	State variables written after the call(s):
	- getPair[token0][token1] = pair (UniswapV2Factory.sol#465)
	- getPair[token1][token0] = pair (UniswapV2Factory.sol#466)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (UniswapV2Factory.sol#384-412):
	External calls:
	- _safeTransfer(_token0,to,amount0Out) (UniswapV2Factory.sol#395)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1Out) (UniswapV2Factory.sol#396)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data)
(UniswapV2Factory.sol#397)
	State variables written after the call(s):
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
		- blockTimestampLast = blockTimestamp (UniswapV2Factory.sol#302)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
		- reserve0 = uint112(balance0) (UniswapV2Factory.sol#300)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
		- reserve1 = uint112(balance1) (UniswapV2Factory.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
UniswapV2Pair.initialize(address,address)._token0 (UniswapV2Factory.sol#284) lacks a zero-check on :
		- token0 = _token0 (UniswapV2Factory.sol#286)
UniswapV2Pair.initialize(address,address)._token1 (UniswapV2Factory.sol#284) lacks a zero-check on :
		- token1 = _token1 (UniswapV2Factory.sol#287)
UniswapV2Factory.constructor(address)._feeToSetter (UniswapV2Factory.sol#442) lacks a zero-check on :
		- feeToSetter = _feeToSetter (UniswapV2Factory.sol#443)
UniswapV2Factory.setFeeTo(address)._feeTo (UniswapV2Factory.sol#471) lacks a zero-check on :
		- feeTo = _feeTo (UniswapV2Factory.sol#473)
UniswapV2Factory.setMigrator(address)._migrator (UniswapV2Factory.sol#476) lacks a zero-check on :
		- migrator = _migrator (UniswapV2Factory.sol#478)
UniswapV2Factory.setFeeToSetter(address)._feeToSetter (UniswapV2Factory.sol#481) lacks a zero-check
on :
		- feeToSetter = _feeToSetter (UniswapV2Factory.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in UniswapV2Pair.burn(address) (UniswapV2Factory.sol#359-381):
	External calls:
	- _safeTransfer(_token0,to,amount0) (UniswapV2Factory.sol#373)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1) (UniswapV2Factory.sol#374)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	State variables written after the call(s):
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
		- price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) *
timeElapsed (UniswapV2Factory.sol#297)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
		- price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) *
timeElapsed (UniswapV2Factory.sol#298)
```

```
Reentrancy in UniswapV2Factory.createPair(address,address) (UniswapV2Factory.sol#454-469):
	External calls:
	- UniswapV2Pair(pair).initialize(token0,token1) (UniswapV2Factory.sol#464)
	State variables written after the call(s):
	- allPairs.push(pair) (UniswapV2Factory.sol#467)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (UniswapV2Factory.sol#384-412):
	External calls:
	- _safeTransfer(_token0,to,amount0Out) (UniswapV2Factory.sol#395)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1Out) (UniswapV2Factory.sol#396)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data)
(UniswapV2Factory.sol#397)
	State variables written after the call(s):
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
		- price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) *
timeElapsed (UniswapV2Factory.sol#297)
	- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
		- price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) *
timeElapsed (UniswapV2Factory.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in UniswapV2Pair.burn(address) (UniswapV2Factory.sol#359-381):
	External calls:
	- _safeTransfer(_token0,to,amount0) (UniswapV2Factory.sol#373)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1) (UniswapV2Factory.sol#374)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	Event emitted after the call(s):
	- Burn(msg.sender,amount0,amount1,to) (UniswapV2Factory.sol#380)
	- Sync(reserve0,reserve1) (UniswapV2Factory.sol#303)
		- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#378)
Reentrancy in UniswapV2Factory.createPair(address,address) (UniswapV2Factory.sol#454-469):
	External calls:
	- UniswapV2Pair(pair).initialize(token0,token1) (UniswapV2Factory.sol#464)
	Event emitted after the call(s):
	- PairCreated(token0,token1,pair,allPairs.length) (UniswapV2Factory.sol#468)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (UniswapV2Factory.sol#384-412):
	External calls:
	- _safeTransfer(_token0,to,amount0Out) (UniswapV2Factory.sol#395)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- _safeTransfer(_token1,to,amount1Out) (UniswapV2Factory.sol#396)
		- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
	- IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data)
(UniswapV2Factory.sol#397)
	Event emitted after the call(s):
	- Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (UniswapV2Factory.sol#411)
	- Sync(reserve0,reserve1) (UniswapV2Factory.sol#303)
		- _update(balance0,balance1,_reserve0,_reserve1) (UniswapV2Factory.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
UniswapV2ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
(UniswapV2Factory.sol#128-140) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(deadline >= block.timestamp,UniswapV2: EXPIRED) (UniswapV2Factory.sol#129)
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (UniswapV2Factory.sol#291-304) uses timestamp
for comparisons
	Dangerous comparisons:
	- timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (UniswapV2Factory.sol#295)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
UniswapV2ERC20.constructor() (UniswapV2Factory.sol#71-85) uses assembly
	- INLINE ASM (UniswapV2Factory.sol#73-75)
UniswapV2Factory.createPair(address,address) (UniswapV2Factory.sol#454-469) uses assembly
	- INLINE ASM (UniswapV2Factory.sol#461-463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['=0.6.12', '>=0.5.0']
	- >=0.5.0 (UniswapV2Factory.sol#9)
	- =0.6.12 (UniswapV2Factory.sol#31)
	- =0.6.12 (UniswapV2Factory.sol#51)
	- =0.6.12 (UniswapV2Factory.sol#145)
	- =0.6.12 (UniswapV2Factory.sol#171)
	- >=0.5.0 (UniswapV2Factory.sol#194)
	- >=0.5.0 (UniswapV2Factory.sol#214)
	- =0.6.12 (UniswapV2Factory.sol#222)
	- =0.6.12 (UniswapV2Factory.sol#430)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.5.0 (UniswapV2Factory.sol#9) allows old versions
Pragma version>=0.5.0 (UniswapV2Factory.sol#194) allows old versions
Pragma version>=0.5.0 (UniswapV2Factory.sol#214) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in UniswapV2Pair._safeTransfer(address,address,uint256) (UniswapV2Factory.sol#262-265):
	- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(UniswapV2Factory.sol#263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable UniswapV2ERC20.DOMAIN_SEPARATOR (UniswapV2Factory.sol#63) is not in mixedCase
Parameter UniswapV2Pair.initialize(address,address)._token0 (UniswapV2Factory.sol#284) is not in
mixedCase
Parameter UniswapV2Pair.initialize(address,address)._token1 (UniswapV2Factory.sol#284) is not in
mixedCase
Parameter UniswapV2Factory.setFeeTo(address)._feeTo (UniswapV2Factory.sol#471) is not in mixedCase
Parameter UniswapV2Factory.setMigrator(address)._migrator (UniswapV2Factory.sol#476) is not in
mixedCase
Parameter UniswapV2Factory.setFeeToSetter(address)._feeToSetter (UniswapV2Factory.sol#481) is not in
mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable UniswapV2Pair.swap(uint256,uint256,address,bytes).balance0Adjusted
(UniswapV2Factory.sol#405) is too similar to
UniswapV2Pair.swap(uint256,uint256,address,bytes).balance1Adjusted (UniswapV2Factory.sol#406)
Variable UniswapV2Pair.price0CumulativeLast (UniswapV2Factory.sol#244) is too similar to
UniswapV2Pair.price1CumulativeLast (UniswapV2Factory.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
UniswapV2Factory.pairCodeHash() (UniswapV2Factory.sol#450-452) uses literals with too many digits:
	- keccak256(bytes)(type()(UniswapV2Pair).creationCode) (UniswapV2Factory.sol#451)
UniswapV2Factory.createPair(address,address) (UniswapV2Factory.sol#454-469) uses literals with too
many digits:
	- bytecode = type()(UniswapV2Pair).creationCode (UniswapV2Factory.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:UniswapV2Factory.sol analyzed (10 contracts with 75 detectors), 37 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> UniswapV2Router02.sol

```
INFO:Detectors:
UniswapV2Router02.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
(UniswapV2Router02.sol#493-509) ignores return value by
IUniswapV2Pair(pair).transferFrom(msg.sender,pair,liquidity) (UniswapV2Router02.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
UniswapV2Library.getAmountsOut(address,uint256,address[]).i (UniswapV2Router02.sol#147) is a local
variable never initialized
UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address).i
(UniswapV2Router02.sol#712) is a local variable never initialized
UniswapV2Router02._swap(uint256[],address[],address).i (UniswapV2Router02.sol#603) is a local variable
never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256)
(UniswapV2Router02.sol#423-450) ignores return value by
IUniswapV2Factory(factory).createPair(tokenA,tokenB) (UniswapV2Router02.sol#433)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
UniswapV2Router02.constructor(address,address)._factory (UniswapV2Router02.sol#413) lacks a
zero-check on :
		- factory = _factory (UniswapV2Router02.sol#414)
UniswapV2Router02.constructor(address,address)._WETH (UniswapV2Router02.sol#413) lacks a
zero-check on :
		- WETH = _WETH (UniswapV2Router02.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
UniswapV2Router02._swap(uint256[],address[],address) (UniswapV2Router02.sol#602-613) has external
calls inside a loop:
IUniswapV2Pair(UniswapV2Library.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new
bytes(0)) (UniswapV2Router02.sol#609-611)
UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address)
(UniswapV2Router02.sol#711-728) has external calls inside a loop: (reserve0,reserve1) = pair.getReserves()
(UniswapV2Router02.sol#719)
UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address)
(UniswapV2Router02.sol#711-728) has external calls inside a loop: amountInput =
IERC20Uniswap(input).balanceOf(address(pair)).sub(reserveInput) (UniswapV2Router02.sol#721)
UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address)
(UniswapV2Router02.sol#711-728) has external calls inside a loop:
pair.swap(amount0Out,amount1Out,to,new bytes(0)) (UniswapV2Router02.sol#726)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['=0.6.12', '>=0.5.0', '>=0.6.0', '>=0.6.2']
	- >=0.5.0 (UniswapV2Router02.sol#9)
	- =0.6.12 (UniswapV2Router02.sol#64)
	- >=0.5.0 (UniswapV2Router02.sol#84)
	- >=0.6.0 (UniswapV2Router02.sol#167)
	- >=0.6.2 (UniswapV2Router02.sol#197)
	- >=0.6.2 (UniswapV2Router02.sol#295)
	- >=0.5.0 (UniswapV2Router02.sol#341)
	- >=0.5.0 (UniswapV2Router02.sol#363)
	- >=0.5.0 (UniswapV2Router02.sol#383)
	- =0.6.12 (UniswapV2Router02.sol#393)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
TransferHelper.safeApprove(address,address,uint256) (UniswapV2Router02.sol#171-175) is never used and
should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.5.0 (UniswapV2Router02.sol#9) allows old versions
Pragma version>=0.5.0 (UniswapV2Router02.sol#84) allows old versions
Pragma version>=0.6.0 (UniswapV2Router02.sol#167) allows old versions
Pragma version>=0.6.2 (UniswapV2Router02.sol#197) allows old versions
Pragma version>=0.6.2 (UniswapV2Router02.sol#295) allows old versions
Pragma version>=0.5.0 (UniswapV2Router02.sol#341) allows old versions
Pragma version>=0.5.0 (UniswapV2Router02.sol#363) allows old versions
Pragma version>=0.5.0 (UniswapV2Router02.sol#383) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (UniswapV2Router02.sol#171-175):
	- (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value))
(UniswapV2Router02.sol#173)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (UniswapV2Router02.sol#177-181):
	- (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value))
(UniswapV2Router02.sol#179)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256)
(UniswapV2Router02.sol#183-187):
	- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value))
(UniswapV2Router02.sol#185)
Low level call in TransferHelper.safeTransferETH(address,uint256) (UniswapV2Router02.sol#189-192):
	- (success) = to.call{value: value}(new bytes(0)) (UniswapV2Router02.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (UniswapV2Router02.sol#26) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (UniswapV2Router02.sol#27) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (UniswapV2Router02.sol#44) is not in mixedCase
Function IUniswapV2Router01.WETH() (UniswapV2Router02.sol#201) is not in mixedCase
Variable UniswapV2Router02.WETH (UniswapV2Router02.sol#406) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tADesired (UniswapV2Router02.sol#206) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (UniswapV2Router02.sol#207)
Variable
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
ADesired (UniswapV2Router02.sol#454) is too similar to
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired
(UniswapV2Router02.sol#427)
Variable
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
ADesired (UniswapV2Router02.sol#454) is too similar to
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
BDesired (UniswapV2Router02.sol#455)
Variable
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired
(UniswapV2Router02.sol#426) is too similar to
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired
(UniswapV2Router02.sol#427)
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tADesired (UniswapV2Router02.sol#206) is too similar to
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired
(UniswapV2Router02.sol#427)
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tADesired (UniswapV2Router02.sol#206) is too similar to
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
BDesired (UniswapV2Router02.sol#455)
```

```
Variable
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
ADesired (UniswapV2Router02.sol#454) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (UniswapV2Router02.sol#207)
Variable
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired
(UniswapV2Router02.sol#426) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (UniswapV2Router02.sol#207)
Variable
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired
(UniswapV2Router02.sol#426) is too similar to
UniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
BDesired (UniswapV2Router02.sol#455)
Variable
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal
(UniswapV2Router02.sol#444) is too similar to
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal
(UniswapV2Router02.sol#439)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
	- UniswapV2Router02.quote(uint256,uint256,uint256) (UniswapV2Router02.sol#793-795)
getAmountOut(uint256,uint256,uint256) should be declared external:
	- UniswapV2Router02.getAmountOut(uint256,uint256,uint256) (UniswapV2Router02.sol#797-805)
getAmountIn(uint256,uint256,uint256) should be declared external:
	- UniswapV2Router02.getAmountIn(uint256,uint256,uint256) (UniswapV2Router02.sol#807-815)
getAmountsOut(uint256,address[]) should be declared external:
	- UniswapV2Router02.getAmountsOut(uint256,address[]) (UniswapV2Router02.sol#817-825)
getAmountsIn(uint256,address[]) should be declared external:
	- UniswapV2Router02.getAmountsIn(uint256,address[]) (UniswapV2Router02.sol#827-835)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UniswapV2Router02.sol analyzed (10 contracts with 75 detectors), 45 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Email: audit@EtherAuthority.io