Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	Unicorn Finance
Website:	unicorn-finance.app
Platform:	Polygonscan
Language:	Solidity
Date:	September 2nd, 2021

Table of contents

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Unicorn Finance team to perform the Security audit of the Unicorn Finance 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 September 2nd, 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

Unicorn Finance's vision is to add a bit of Magic to DeFi by enabling investors to take their crypto investments to the moon with a mix of spellbinding investment strategies conjured to deliver growth and long term value for the platform and customer while keeping users' funds SAFU.

Audit scope

Name	Code Review and Security Analysis Report for Unicorn Finance Smart Contract
Platform	Polygonscan / Solidity
File 1	CswapUniSWAP.sol
File 1 MD5 Hash	431BC9F5F83637DE9B935FD7FD24D39A
Updated File 1 MD5 Hash	B0AA2ABF6A55DED7141A9436694B53A9
File 2	UnicornMasterChef.sol
File 2 MD5 Hash	27F7C49A0F07EDB6A8CAF521BC2BB163
Updated File 2 MD5 Hash	315B0407F38651A24FC2DA1B90DFB4BA
File 3	UnicornPresale.sol
File 3 MD5 Hash	BEC1D2756B86202347B0036A0C2E1BC8

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Updated File 3 MD5 Hash	338DA6522AF6C356E495552B77C960D0
File 4	UnicornReferral.sol
File 4 MD5 Hash	9F2ECAE8704A22F66B62A0FE15B9E395
File 5	UnicornToken.sol
File 5 MD5 Hash	04F72B39E89416FE6476141CB2367AA6
File 6	Timelock.sol
File 6 MD6 Hash	4E725A5AA078C8259BF28DF590E6B96D
Audit Date	September 2nd, 2021
Revised Audit Date	September 6th, 2021

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1: CswapUniSWAP.sol • The CswapuniSTWAP owner can access functions like: withdrawTokens, withdrawBalance, getCswapBalance, swapCWAP etc.	**YES, This is valid.**
File 2: UnicornMasterChef.sol • Referral Commission Rate: 1% • Maximum Referral Commission Rate: 10% • Maximum Fee: 4%	**YES, This is valid.**
File 3: UnicornPresale.sol • Maximum buy Limit: 1000	**YES, This is valid.**
File 4: UnicornReferral.sol • The UnicornReferral owner can access functions like: recordReferral, updateOperator, recordReferralCommission, etc.	**YES, This is valid.**
File 5: UnicornToken.sol • Name: Unicorn • Symbol: UNIQ • Decimals: 18 • LP Tax Fee: 1% • Project Tax Fee: 4% • Maximum Supply: 0.5 million	**YES, This is valid.**
File 6: Timelock.sol • Grace Period: 14 days • Minimum Delay: 6 hours	**YES, This is valid.**

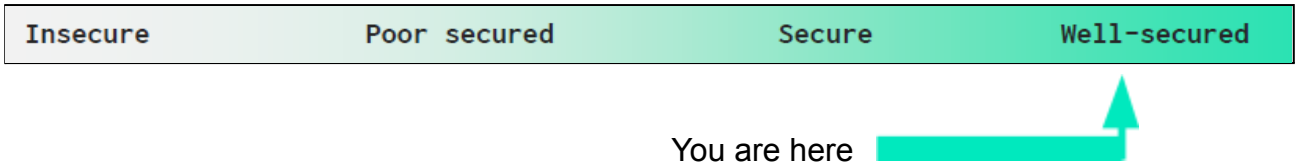
This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

• Maximum Delay: 30 days	

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does not contain any owner control, making it fully decentralized.



We used various tools like Slither, Solhint 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 11 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.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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	Passed
	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
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	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**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Code Quality

This audit scope has 6 smart contract files. Smart contracts also contain Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

The libraries in Unicorn Finance 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 Unicorn Finance .

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

Code parts are **not well** commented on smart contracts.

Documentation

We were given a Unicorn Finance smart contracts code in the form of a github.com web link. The hashes of that code are mentioned above in the table.

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

Another source of information was its official website https://unicorn-finance.app/ which provided rich information about the project architecture and tokenomics.

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 not used in external smart contract calls.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

AS-IS overview

CswapUniSWAP.sol

(1) Interface

(a) IERC20

(2) Usages

(a) using SafeMath for uint256;

(b) using SafeERC20 for IERC20;

(3) Events

(a) event CSWAPExchanged(address user, uint256 amount);

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	IsOwner	modifier	Passed	No Issue
3	changeOwner	write	Passed	No Issue
4	setAddresses	write	Passed	No Issue
5	safeTransferUnicorn	internal	Passed	No Issue
6	swapCWAP	write	Passed	No Issue
7	getUnicornBalance	read	Passed	No Issue
8	getCswapBalance	read	Passed	No Issue
9	withdrawTokens	write	Passed	No Issue
10	withdrawBalance	write	Passed	No Issue

UnicornMasterChef.sol

(1) Interface

(a) IUnicornReferral

(2) Inherited contracts

(a) Ownable

(b) Context

(c) UnicornToken

(3) Usages

(a) using SafeMath for uint256;

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

(b) using SafeBEP20 for IBEP20;

(4) Struct

(a) UserInfo

(b) PoolInfo

(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 ReferralCommissionPaid(address indexed user, address indexed referrer, uint256 commissionAmount);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	poolLength	external	Passed	No Issue
3	setReferralCommissionRate	write	access only Owner	No Issue
4	add	write	Critical operation lacks event log	Refer audit findings section below
5	set	write	Critical operation lacks event log	Refer audit findings section below
6	getMultiplier	write	Passed	No Issue
7	pendingUNIQ	external	Passed	No Issue
8	setUnicornReferral	write	Critical operation lacks event log	Refer audit findings section below
9	massUpdatePools	write	Infinite loops possibility at multiple places	Refer audit findings section below
10	updatePool	write	Critical operation lacks event log	Refer audit findings section below
11	deposit	write	Passed	No Issue
12	withdraw	write	Passed	No Issue
13	emergencyWithdraw	write	Function input parameters lack of check	Refer audit findings section below
14	safeUNIQTransfer	internal	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

15	dev	write	Critical operation lacks event log	Refer audit findings section below
16	setFeeAddress	write	Critical operation lacks event log	Refer audit findings section below
17	updateEmissionRate	write	Critical operation lacks event log	Refer audit findings section below
18	payReferralCommission	internal	Function input parameters lack of check	Refer audit findings section below
19	owner	read	Passed	No Issue
20	onlyOwner	modifier	Passed	No Issue
21	renounceOwnership	write	access only Owner	No Issue
22	transferOwnership	write	access only Owner	No Issue

UnicornPresale.sol

(1) Interface

(a) IERC20

(2) Usages

(a) using SafeMath for uint256;

(3) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	IsOwner	modifier	Passed	No Issue
3	changeOwner	write	Passed	No Issue
4	setPresaleStarts	write	access by IsOwner	No Issue
5	changePrice	write	Passed	No Issue
6	safeTransferTokens	internal	Passed	No Issue
7	setMaxLimit	write	Operation lacks event log	Refer audit findings section below
8	buyToken	write	Passed	No Issue
9	calculateTokens	read	Passed	No Issue
10	getTokenBalance	read	Passed	No Issue
12	withdrawTokens	write	Passed	No Issue
13	withdrawBalance	write	Passed	No Issue
14	changePresaleEndTime	write	Operation lacks event log	Refer audit findings section below

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

UnicornReferral.sol

(1) Interface

(a) IUnicornReferral

(b) IBEP20

(2) Inherited contracts

(a) Ownable

(b) IUnicornReferral

(c) Context

(3) Usages

(a) using SafeBEP20 for IBEP20;

(4) Events

(a) event ReferralRecorded(address indexed user, address indexed referrer);

(b) event ReferralCommissionRecorded(address indexed referrer, uint256 commission);

(c) event OperatorUpdated(address indexed operator, bool indexed status);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	recordReferral	write	Other code specification issues	Refer audit findings section below
2	onlyOperator	modifier	Passed	No Issue
3	recordReferralCommission	write	Other code specification issues	Refer audit findings section below
4	getReferrer	read	Passed	No Issue
5	updateOperator	external	access only Operator	No Issue
6	constructor	write	Passed	No Issue
7	owner	read	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	renounceOwnership	write	access only Owner	No Issue
10	transferOwnership	write	access only Owner	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

UnicornToken.sol

(1) Interface

(a) IBEP20

(b) IUniswapV2Factory

(c) IUniswapV2Pair

(d) IUniswapV2Router01

(e) IUniswapV2Router02

(2) Inherited contracts

(a) Ownable

(b) Context

(3) Usages

(a) using SafeMath for uint256;

(4) Events

(a) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);

(b) event SwapAndLiquifyEnabledUpdated(bool enabled);

(c) event SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiqudity);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOperator	modifier	Passed	No Issue
3	changeProjectWallet	write	Function input parameters lack of check	Refer audit findings section below
4	setNumTokensSellToAddToLiquidity	write	Function input parameters lack of check	Refer audit findings section below
5	addOrRemoveFromWhitelist	write	Function input parameters lack of check	Refer audit findings section below
6	setOperator	write	Function input parameters lack of check	Refer audit findings section below
7	enableDisableFee	write	access only Operator	No Issue
8	setLiquification	write	access only Operator	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

9	mint	write	access only Owner	No Issue
10	name	read	Passed	No Issue
11	symbol	read	Passed	No Issue
12	decimals	read	Passed	No Issue
13	totalSupply	read	Passed	No Issue
14	balanceOf	read	Passed	No Issue
15	transfer	write	Passed	No Issue
16	allowance	read	Passed	No Issue
17	approve	write	Passed	No Issue
18	transferFrom	write	Passed	No Issue
19	increaseAllowance	write	Passed	No Issue
20	decreaseAllowance	write	Passed	No Issue
21	mint	internal	access only Owner	No Issue
22	_transfer	internal	Passed	No Issue
23	_mint	internal	Passed	No Issue
24	_burn	internal	Passed	No Issue
25	_approve	internal	Passed	No Issue
26	_burnFrom	internal	Passed	No Issue
27	swapAndLiquify	write	Passed	No Issue
28	swapTokensForEth	write	Passed	No Issue
29	addLiquidity	write	Tokens goes to dead wallet address	Refer audit findings section below
30	owner	read	Passed	No Issue
31	onlyOwner	modifier	Passed	No Issue
32	renounceOwnership	write	access only Owner	No Issue
33	transferOwnership	write	access only Owner	No Issue

Timelock.sol

(1) Usages

(a) using SafeMath for uint;

(2) Event

(a) event NewAdmin(address indexed newAdmin);

(b) event NewPendingAdmin(address indexed newPendingAdmin);

(c) event NewDelay(uint indexed newDelay);

(d) event CancelTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

(e) event ExecuteTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

(f) event QueueTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

(3) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	receive	external	Passed	No Issue
3	setDelay	write	Passed	No Issue
4	acceptAdmin	write	Passed	No Issue
5	setPendingAdmin	write	Passed	No Issue
6	queueTransaction	write	Passed	No Issue
7	cancelTransaction	write	Passed	No Issue
8	executeTransaction	write	Passed	No Issue
9	getBlockTimestamp	internal	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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
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.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Findings

CswapUniSWAP.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

```
function changeOwner (address addr) public {
    require(msg.sender == owner,"You are not authorized");

    owner = addr;
}
```

```
function  setAddresses(address _cswap,address _unicorn) public {
    require(msg.sender == owner,"You are not authorized");
    unicorn= IERC20(_unicorn);
    cswap = IERC20(_cswap);
}
```

```
function swapCWAP(uint256 amount) public  {
    cswap.safeTransferFrom(msg.sender,address(this),amount);
    safeTransferUnicorn(amount);
    emit CSWAPExchanged(msg.sender,amount);
}
```

There are many functions which have not been validated.

Resolution: Put validation:

Function changeOwner() -> addr -> variable value is not address(0)

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Function setAddresses() -> _cswap, _unicorn -> variable value is not address(0)

Function swapCWAP() -> amount, -> variable value is not less than or equal 0.

Status: Fixed.

(2) Owner can drain tokens and balance

```
// use this fuction for withdrawing all the unsold tokens

function withdrawTokens( ) public{
    require(msg.sender == owner,"You are not the owner");
    cswap.safeTransfer(owner,getUnicornBalance());
    unicorn.safeTransfer(owner,getCswapBalance());

}



// use this fuction for withdrawing all the ethers
function withdrawBalance( ) public{
    require(msg.sender == owner,"You are not the owner");
    payable(msg.sender).transfer(address(this).balance);
}
```

By these functions, the owner can drain all the tokens and contract balance.

Resolution: The owner's wallet private key would be compromised, then it would create trouble. If it is a part of the plan then disregard.

Status: Acknowledged by auditee.

Comments: Pre-sale will work in the way that the tokens can be distributed from the owner's wallet to the investor. Ownership of masterchef will be transferred to timelock and there won't be any changes possible.

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Status: **Acknowledged by auditee.**

(2) Warning: SPDX license identifier:

```
CswapUniSWAP.sol: Warning: SPDX license identifier not provided in source
file. Before publishing, consider adding a comment containing "SPDX-License-
Identifier: <SPDX-License>" to each source file. Use "SPDX-License-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org
for more information.
```

SafeMath.sol: Warning: SPDX license identifier not provided in source file.

Resolution: SPDX-License-Identifier.

Status: **Acknowledged by auditee.**

(3) Unwanted spaces:

```
contract CswapUniSWAP{
    using SafeMath for uint256;
    using SafeERC20 for IERC20;




    event CSWAPExchanged(address user, uint256 amount);

    address public owner;
    IERC20 public cswap;
    IERC20 public unicorn;


    uint256 public totalCSWAPExchanged;









    constructor() public{
        owner = msg.sender;

    }


    modifier IsOwner{
        require(msg.sender == owner);
        _;
    }
```

There are many unwanted spaces and break lines.

Resolution: Remove unwanted spaces and break lines.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Status: **Fixed.**

(4) Comment Typo:

```
// use this fuction for withdrawing all the unsold tokens

function withdrawTokens( ) public{
```

```
// use this fuction for withdrawing all the ethers
function withdrawBalance( ) public{
```

The linked comment statement contains a typo in its body, namely "fuction".

Resolution: We advise that the comment text is corrected.

Status: **Fixed.**

(5) Unused modifier:

```
modifier IsOwner{
    require(msg.sender == owner);
    _;
}
```

Resolution: We advise to remove the unused modifier.

Status: **Fixed.**

UnicornMasterChef.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

The missing event log for :

- add() - onlyOwner
- set() - onlyOwner
- setUnicornReferral() - onlyOwner
- massUpdatePools()
- updatePool()
- dev()
- setFeeAddress()
- updateEmissionRate()

Resolution: Please write an event log for listed events.

Status: **Acknowledged by auditee.**

(2) Infinite loops possibility at multiple places:

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

There is a function massUpdatePools() in the smart contracts, where the poolInfo.length variable is used directly in the loop. It is recommended to put some kind of limits.

Resolution: So it does not go wild and create any scenario where it can hit the block gas limit.

Status: **Acknowledged by auditee.**

Comments: Pools are supposed to be limited and hence kept it that way.

(3) Function input parameters lack of check:

There are many functions which have not been validated.

Resolution: Put validation:

- Function payReferralCommission() -> _pending -> value is not less than or equal 0
- Function payReferralCommission() -> _user -> variable value is not address(0)
- Function setFeeAddress() -> _feeAddress -> variable value is not address(0)

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

- Function emergencyWithdraw() -> _pid -> value is not less than or equal 0

Status: **Acknowledged by auditee.**

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: **Acknowledged by auditee.**

(2) Make variable constant

```
uint256 private MAX_FEE = 400; //4%
```

These variable's' values will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare this variable as constant. Just put a constant keyword. And define constants in the constructor.

Status: **Fixed.**

UnicornPresale.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
function changeOwner (address addr) public {
    require(msg.sender == owner,"You are not authorized");

    owner = addr;
}
```

```
function changePrice(uint256 price) public {
    require(msg.sender == owner,"You are not authorized");
    _price = price;
}
```

There are many functions which have not been validated.

Resolution: Put validation:

- Function changeOwner() -> addr -> variable value is not address(0)
- Function changePrice() -> price -> variable value is not less than or equal 0

Status: **Fixed.**

(2) Operation lacks event log:

```
function changePrice(uint256 price) public {
    require(msg.sender == owner,"You are not authorized");
    _price = price;
}
```

```
function changeOwner (address addr) public {
    require(msg.sender == owner,"You are not authorized");

    owner = addr;
}
```

```
function setMaxLimit(uint256 maxL) public {
    require(msg.sender == owner,"You are not authorized");

    MAX_BUY_LIMIT =  maxL;
}
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
function changePresaleEndTime(uint256 time) public{
    require(msg.sender == owner,"You are not the owner");
    presaleTimeEnds = time;

}
```

The missing event log for :

- changeOwner()
- changePrice()
- setMaxLimit()
- changePresaleEndTime()

Resolution: Please write an event log for listed events.

Status: **Acknowledged by auditee.**

(3) Owner can drain tokens and balance

```
// use this fuction for withdrawing all the unsold tokens

function withdrawTokens( ) public{
    require(msg.sender == owner,"You are not the owner");
    token.transfer(owner,getTokenBalance());

}



// use this fuction for withdrawing all the ethers
function withdrawBalance( ) public{
    require(msg.sender == owner,"You are not the owner");
    payable(msg.sender).transfer(getContractBalance());
}
```

By these functions, the owner can drain all the tokens and contract balance.

Resolution: The owner's wallet private key would be compromised, then it would create trouble. If it is a part of the plan then disregard.

Status: **Acknowledged by auditee.**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Comments: Pre-sale will work in the way that the tokens can be distributed from the owner's wallet to the investor. Ownership of masterchef will be transferred to timelock and there won't be any changes possible.

(4) Wrong error message

```
function buyToken() public payable {
    require(presaleTimeEnds > block.timestamp,"Presale Finished");

    require(presaleStarts < block.timestamp,"Presale not started");




    uint256 noOfTokens = calculateTokens(msg.value);
    uint256 preBalance = token.balanceOf(msg.sender);

    require(noOfTokens.add(preBalance)<= MAX_BUY_LIMIT,"You can't have more than 2000 tokens");
    require(noOfTokens<= MAX_BUY_LIMIT,"You can't buy more than 2000 tokens");

    safeTransferTokens(noOfTokens);
}
```

MAX_BUY_LIMIT sets 1000 as default and can be modified in future. But the error message shows static value 2000 tokens as buy limit.

Resolution: We suggest changing the error message.

Status: **Acknowledged by auditee.**

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: **Acknowledged by auditee.**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

(2) Unwanted spaces:

```
contract UnicornPresale{
    using SafeMath for uint256;




    address public owner;
    IERC20 public token;
    uint256 public _price;

    uint256 public presaleTimeEnds;
    uint256 public totalTokenSold;
    uint256 public presaleStarts;

    uint256 public MAX_BUY_LIMIT = 1000*1e18;




    uint256 public normalSaleSold = 0;
```

There are many unwanted spaces and break lines.

Resolution: Remove unwanted spaces and break lines.

Status: **Acknowledged by auditee.**

(3) Comment Typo:

```
// this fucntion is used to check how many fokitos are remaining in the contract
function getTokenBalance() public view returns(uint256){
```

```
    // this fucntion is used to check how many ethers are there in the contract

function getContractBalance() public view returns(uint256){
```

```
// use this fuction for withdrawing all the unsold tokens

function withdrawTokens( ) public{
```

```
// use this fuction for withdrawing all the ethers
function withdrawBalance( ) public{
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

The linked comment statement contains a typo in its body, namely: (fuction, fucntion, fokitos)

Resolution: We advise that the comment text is corrected.

Status: Fixed.

(4) Unused variable:

```
uint256 public normalSaleSold = 0;
```

There are unused variables.

Resolution: Remove unused variables.

Status: Fixed.

(5) Missing error message

```
modifier IsOwner{
    require(msg.sender == owner);
    _;
}
```

When the required condition fails there should be a message to show the error or cause of the failed transaction.

Resolution: add relevant error message to require conditions.

Status: Fixed.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

(6) Not used defined modifier

```
modifier IsOwner{
    require(msg.sender == owner);
    _;
}



function changeOwner (address addr) public {
    require(msg.sender == owner,"You are not authorized");

    owner = addr;
}
function changePrice(uint256 price) public {
    require(msg.sender == owner,"You are not authorized");
    _price = price;
}
```

There is a modifier defined in code and at some places the same validation has been done as in the modifier.

Resolution: We suggest using a predefined modifier.

Status: **Fixed.**

UnicornReferral.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Low

(1) Required 'require' statement

```
function recordReferral(address _user, address _referrer) public override onlyOperator {
    if (_user != address(0)
        && _referrer != address(0)
        && _user != _referrer
        && referrers[_user] == address(0)
    ) {
        referrers[_user] = _referrer;
        referralsCount[_referrer] += 1;
        emit ReferralRecorded(_user, _referrer);
    }
}

function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
    if (_referrer != address(0) && _commission > 0) {
        totalReferralCommissions[_referrer] += _commission;
        emit ReferralCommissionRecorded(_referrer, _commission);
    }
}
```

Resolution:In these functions, if condition should be replaced by require statement to stop the transaction if not matched with the conditions.

Status: **Acknowledged by auditee.**

Comments: These functions will be called by the master chef. So can't use require statement.

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: **Acknowledged by auditee.**

(2) Other code specification issues:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
function recordReferral(address _user, address _referrer) public override onlyOperator {
    if (_user != address(0)
        && _referrer != address(0)
        && _user != _referrer
        && referrers[_user] == address(0)
    ) {
        referrers[_user] = _referrer;
        referralsCount[_referrer] += 1;
        emit ReferralRecorded(_user, _referrer);
    }
}
```

```
function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
    if (_referrer != address(0) && _commission > 0) {
        totalReferralCommissions[_referrer] += _commission;
        emit ReferralCommissionRecorded(_referrer, _commission);
    }
}
```

SafeMath Library imported but not used

Resolution: Use SafeMath Library.

Status: **Acknowledged by auditee.**

UnicornToken.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

```
function changeProjectWallet(address addr) public onlyOperator{
    projectWallet = addr;
}
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
function setNumTokensSellToAddToLiquidity(uint256 limit) public onlyOperator{
    numTokensSellToAddToLiquidity = limit;
}
```

```
function addOrRemoveFromWhitelist(address addr,bool isAdd) public onlyOperator{
    whitelist[addr] = isAdd;
}
```

```
function setOperator(address newOperator) public onlyOperator{
    operator = newOperator;
}
```

There are many functions which have not been validated.

Resolution: Put validation:

- Function changeProjectWallet() -> addr -> variable value is not address(0)
- Function setNumTokensSellToAddToLiquidity() -> limit -> variable value is not less than or equal 0
- Function addOrRemoveFromWhitelist() -> addr -> variable value is not address(0)
- Function setOperator() -> newOperator -> variable value is not address(0)

Status: **Acknowledged by auditee.**

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: **Acknowledged by auditee.**

(2) Make variables constant:

```
uint256 public projectTax = 40; //4%
uint256 public lpTax = 10; //1%
```

```
uint256 public MAX_SUPPLY = 500000*1e18;
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

These variables' values will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare those variables as constant. Just put a constant keyword. And define constants in the constructor.

Status: **Acknowledged by auditee.**

(3) Tokens goes to dead wallet address

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        address(0x000000000000000000000000000000000000dEaD),
        block.timestamp
    );
}
```

In addLiquidity function, tokens get burned by sending to the dead wallet address.

Resolution: If it is a part of the plan then disregard.

Status: **Acknowledged by auditee.**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Centralization

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

- setReferralCommissionRate: The UnicornMasterChef owner can update referral commission rate.
- add: The UnicornMasterChef owner can add a pool.
- set: The UnicornMasterChef owner can update the given pool's UNIQ allocation point and deposit fee.
- setUnicornReferral: Update the panther referral contract address by the UnicornMasterChef owner.
- updateEmissionRate: The UnicornMasterChef owner can add hidden dummy pools in order to alter the emission, here we make it simple and transparent to all.
- setPresaleStarts: The UnicornPresale owner can set presaleStarts time.
- recordReferral: The Operator owner can save a record referral address.
- recordReferralCommission: The Operator owner can save record Referral Commission.
- updateOperator: The Operator owner can update the status of the operator.
- changeProjectWallet: The Operator owner can change the project wallet address.
- setNumTokensSellToAddToLiquidity: The Operator owner can set number tokens set to add liquidity limit.
- addOrRemoveFromWhitelist: The Operator owner can add or remove whitelist addresses.
- setOperator: The Operator owner can set operator addresses.
- enableDisableFee: The Operator owner can set enable disable fee.
- setLiquification: The Operator owner can set liquefaction status.
- mint: The Unicorn token Owner can set a mint amount.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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 issues are not critical ones. So, **it's good to go to 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".**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

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

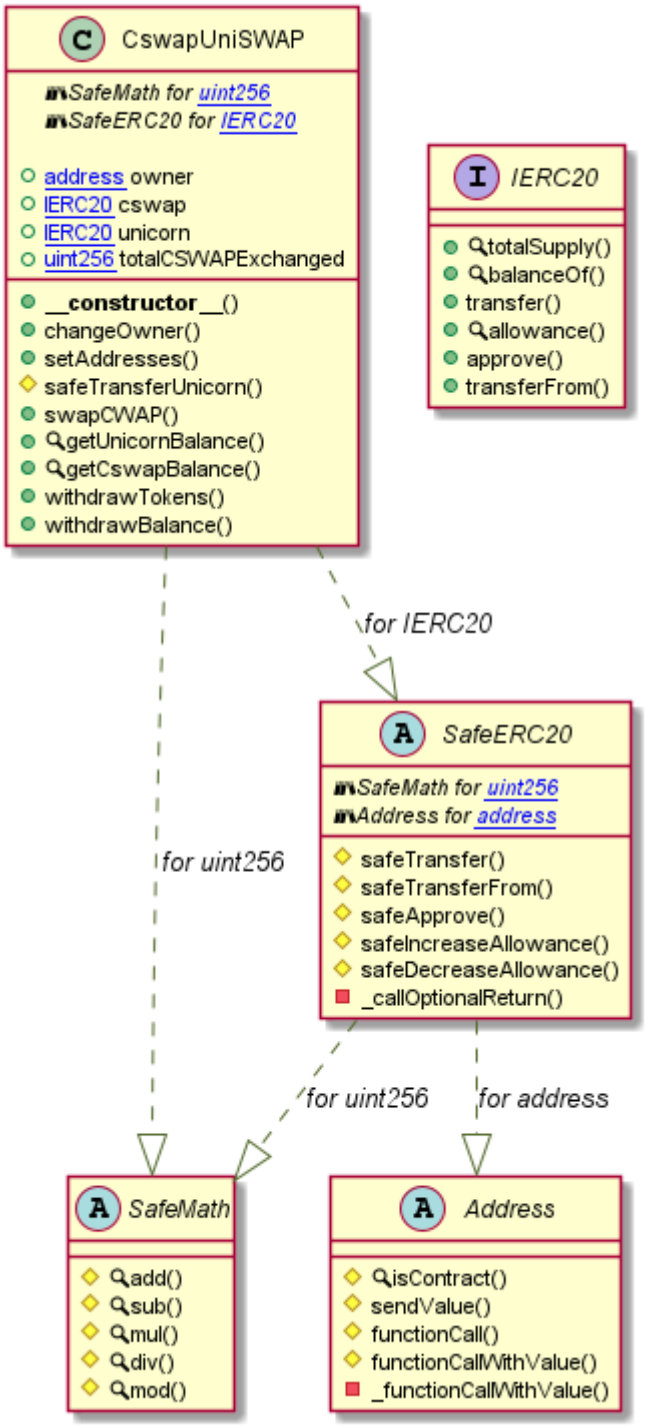
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Appendix

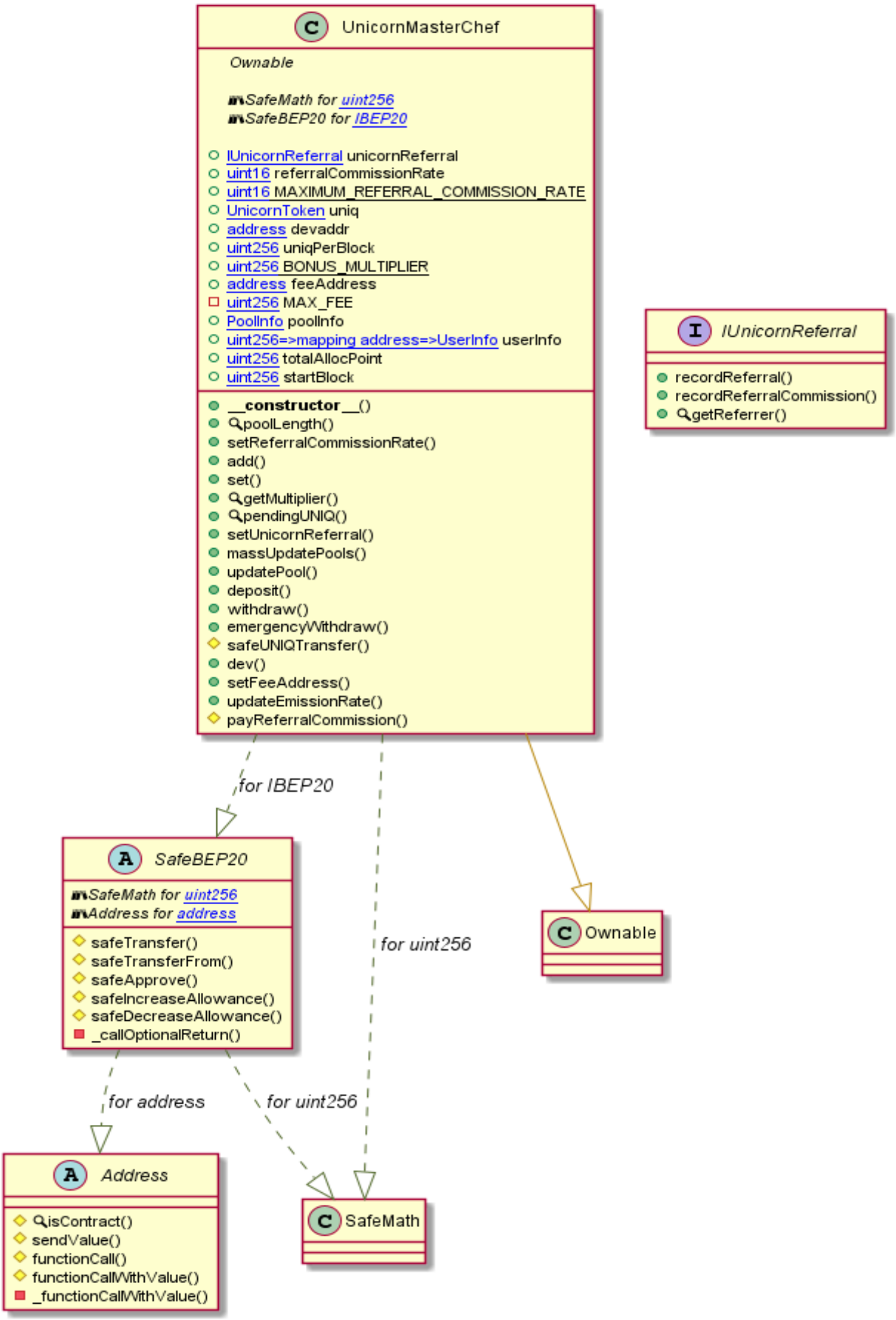
Code Flow Diagram - Unicorn Protocol

CswapUniSWAP Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

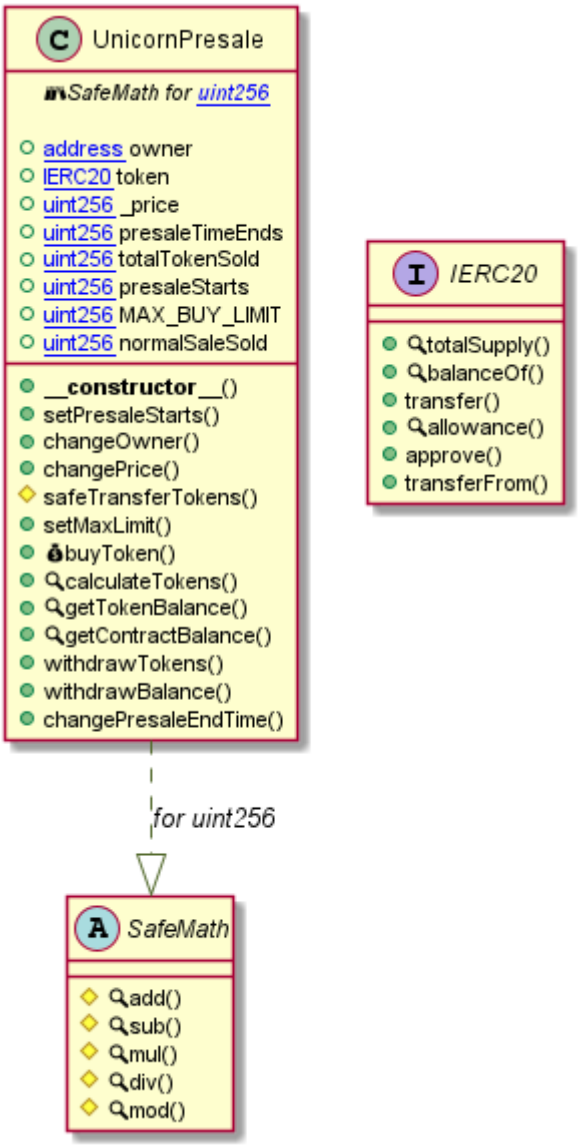
UnicornMasterChef Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

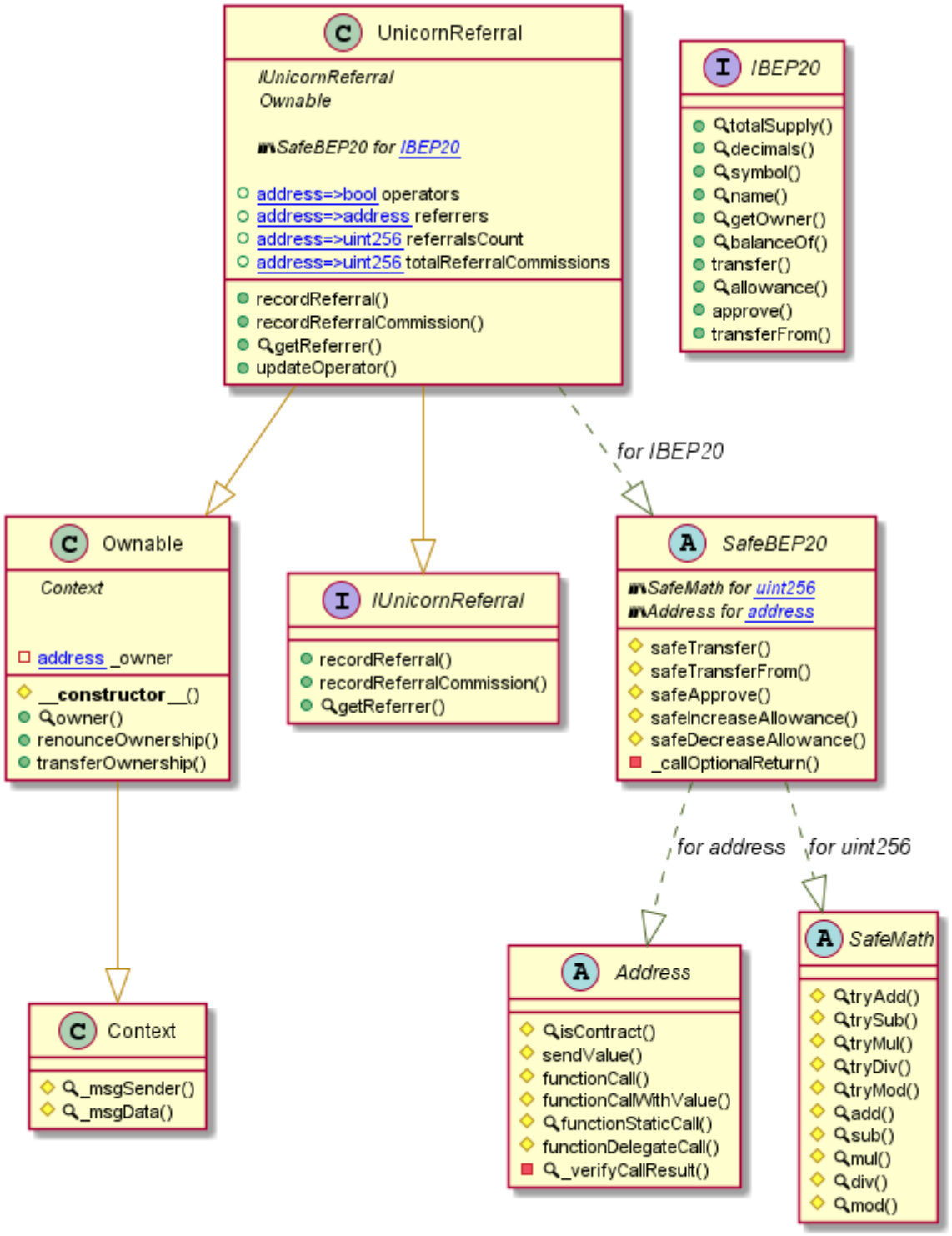
Email: audit@EtherAuthority.io

UnicornPresale Diagram



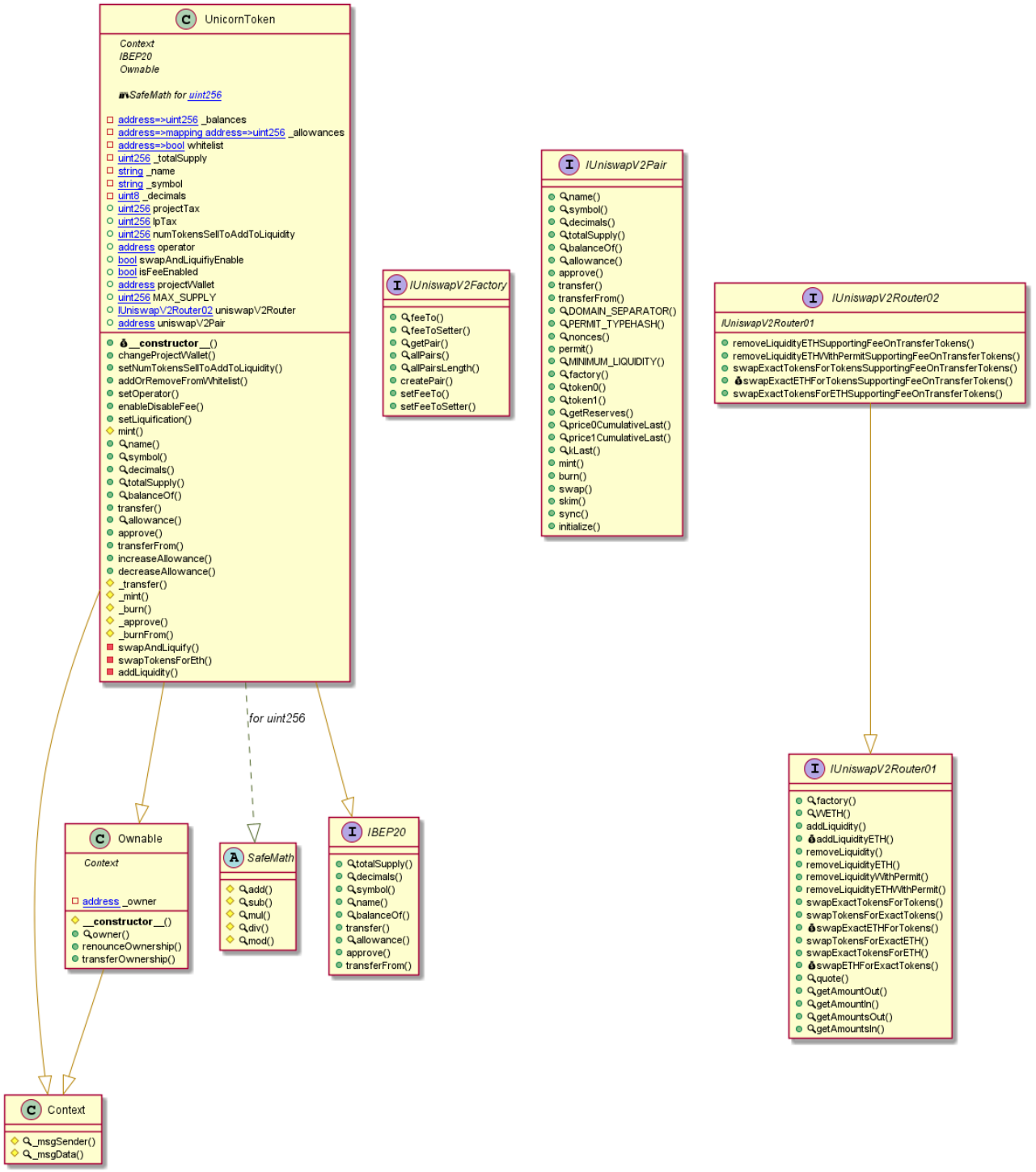
This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

UnicornReferral Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

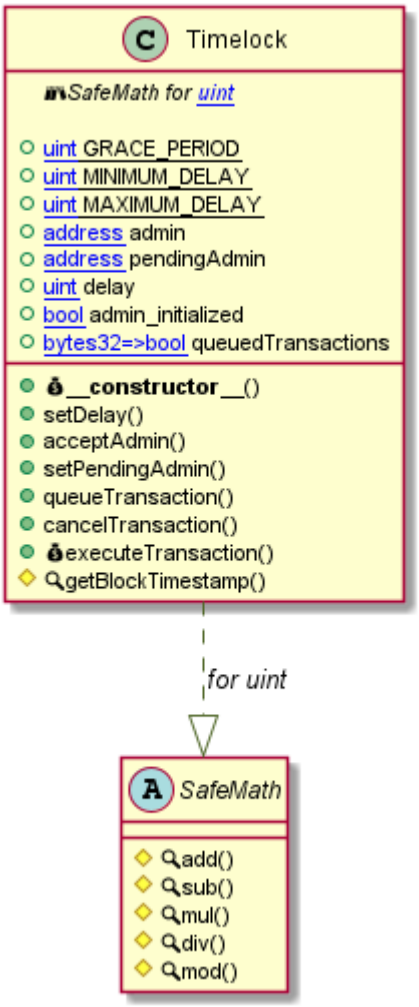
UnicornToken Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Timelock Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> CswapUniSWAP.sol

```
INFO:Detectors:
CswapUniSWAP.changeOwner(address) (CswapUniSWAP.sol#444-448) should emit an event for:
        - owner = addr (CswapUniSWAP.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
CswapUniSWAP.changeOwner(address).addr (CswapUniSWAP.sol#444) lacks a zero-check on :
                - owner = addr (CswapUniSWAP.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CswapUniSWAP.safeTransferUnicorn(uint256) (CswapUniSWAP.sol#465-469):
        External calls:
        - unicorn.safeTransfer(msg.sender,noOfTokens) (CswapUniSWAP.sol#467)
        State variables written after the call(s):
        - totalCSWAPExchanged = totalCSWAPExchanged.add(noOfTokens) (CswapUniSWAP.sol#468)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CswapUniSWAP.swapCWAP(uint256) (CswapUniSWAP.sol#477-481):
        External calls:
        - cswap.safeTransferFrom(msg.sender,address(this),amount) (CswapUniSWAP.sol#478)
        - safeTransferUnicorn(amount) (CswapUniSWAP.sol#479)
                - unicorn.safeTransfer(msg.sender,noOfTokens) (CswapUniSWAP.sol#467)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CswapUniSWAP.sol#387)
                - (success,returndata) = target.call{value: weiValue}(data) (CswapUniSWAP.sol#142)
        External calls sending eth:
        - safeTransferUnicorn(amount) (CswapUniSWAP.sol#479)
                - (success,returndata) = target.call{value: weiValue}(data) (CswapUniSWAP.sol#142)
        Event emitted after the call(s):
        - CSWAPExchanged(msg.sender,amount) (CswapUniSWAP.sol#480)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (CswapUniSWAP.sol#25-36) uses assembly
        - INLINE ASM (CswapUniSWAP.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (CswapUniSWAP.sol#133-159) uses assembly
        - INLINE ASM (CswapUniSWAP.sol#151-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (CswapUniSWAP.sol#80-82) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CswapUniSWAP.sol#109-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (CswapUniSWAP.sol#123-131) is never used and should be removed
Address.sendValue(address,uint256) (CswapUniSWAP.sol#54-60) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (CswapUniSWAP.sol#339-353) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (CswapUniSWAP.sol#364-374) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (CswapUniSWAP.sol#355-362) is never used and should be removed
SafeMath.div(uint256,uint256) (CswapUniSWAP.sol#244-246) is never used and should be removed
SafeMath.div(uint256,uint256,string) (CswapUniSWAP.sol#259-266) is never used and should be removed
SafeMath.mod(uint256,uint256) (CswapUniSWAP.sol#279-281) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CswapUniSWAP.sol#294-297) is never used and should be removed
SafeMath.mul(uint256,uint256) (CswapUniSWAP.sol#219-231) is never used and should be removed
SafeMath.sub(uint256,uint256) (CswapUniSWAP.sol#190-192) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (CswapUniSWAP.sol#203-208) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (CswapUniSWAP.sol#54-60):
        - (success) = recipient.call{value: amount}() (CswapUniSWAP.sol#58)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (CswapUniSWAP.sol#133-159):
        - (success,returndata) = target.call{value: weiValue}(data) (CswapUniSWAP.sol#142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter CswapUniSWAP.setAddresses(address,address)._cswap (CswapUniSWAP.sol#456) is not in mixedCase
Parameter CswapUniSWAP.setAddresses(address,address)._unicorn (CswapUniSWAP.sol#456) is not in mixedCase
Modifier CswapUniSWAP.IsOwner() (CswapUniSWAP.sol#436-439) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
changeOwner(address) should be declared external:
        - CswapUniSWAP.changeOwner(address) (CswapUniSWAP.sol#444-448)
setAddresses(address,address) should be declared external:
        - CswapUniSWAP.setAddresses(address,address) (CswapUniSWAP.sol#456-460)
swapCWAP(uint256) should be declared external:
        - CswapUniSWAP.swapCWAP(uint256) (CswapUniSWAP.sol#477-481)
withdrawTokens() should be declared external:
        - CswapUniSWAP.withdrawTokens() (CswapUniSWAP.sol#509-514)
withdrawBalance() should be declared external:
        - CswapUniSWAP.withdrawBalance() (CswapUniSWAP.sol#519-522)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CswapUniSWAP.sol analyzed (5 contracts with 75 detectors), 30 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Slither log >> UnicornMasterChef.sol

```
INFO:Detectors:
UnicornToken.addLiquidity(uint256,uint256) (UnicornMasterChef.sol#1164-1177) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000000000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
UnicornMasterChef.safeUNIQTransfer(address,uint256) (UnicornMasterChef.sol#1427-1434) ignores return value by uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
UnicornMasterChef.safeUNIQTransfer(address,uint256) (UnicornMasterChef.sol#1427-1434) ignores return value by uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
UnicornMasterChef.pendingUNIQ(uint256,address) (UnicornMasterChef.sol#1314-1325) performs a multiplication on the result of a division:
        -uniqReward = multiplier.mul(uniqPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (UnicornMasterChef.sol#1321)
        -accUNIQPerShare = accUNIQPerShare.add(uniqReward.mul(1e12).div(lpSupply)) (UnicornMasterChef.sol#1322)
UnicornMasterChef.updatePool(uint256) (UnicornMasterChef.sol#1342-1358) performs a multiplication on the result of a division:
        -uniqReward = multiplier.mul(uniqPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (UnicornMasterChef.sol#1353)
        -pool.accUNIQPerShare = pool.accUNIQPerShare.add(uniqReward.mul(1e12).div(lpSupply)) (UnicornMasterChef.sol#1356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
UnicornMasterChef.updatePool(uint256) (UnicornMasterChef.sol#1342-1358) uses a dangerous strict equality:
        - lpSupply == 0 || pool.allocPoint == 0 (UnicornMasterChef.sol#1348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in UnicornMasterChef.add(uint256,IBEP20,uint16,bool) (UnicornMasterChef.sol#1269-1290):
        External calls:
        - massUpdatePools() (UnicornMasterChef.sol#1277)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP)) (UnicornMasterChef.sol#1281-1289)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (UnicornMasterChef.sol#1280)
Reentrancy in UnicornMasterChef.deposit(uint256,uint256,address) (UnicornMasterChef.sol#1361-1392):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1364)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
```

```
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - unicornReferral.recordReferral(msg.sender,_referrer) (UnicornMasterChef.sol#1367)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1372)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        - safeUNIQTransfer(msg.sender,pending.sub(comissionPaid)) (UnicornMasterChef.sol#1373)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (UnicornMasterChef.sol#1378)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (UnicornMasterChef.sol#1384)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (UnicornMasterChef.sol#1385)
        - user.rewardDebt = user.amount.mul(pool.accUNIQPerShare).div(1e12) (UnicornMasterChef.sol#1390)
Reentrancy in UnicornMasterChef.deposit(uint256,uint256,address) (UnicornMasterChef.sol#1361-1392):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1364)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - unicornReferral.recordReferral(msg.sender,_referrer) (UnicornMasterChef.sol#1367)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1372)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        - safeUNIQTransfer(msg.sender,pending.sub(comissionPaid)) (UnicornMasterChef.sol#1373)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (UnicornMasterChef.sol#1378)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (UnicornMasterChef.sol#1387)
Reentrancy in UnicornMasterChef.set(uint256,uint256,uint16,bool) (UnicornMasterChef.sol#1293-1306):
        External calls:
        - massUpdatePools() (UnicornMasterChef.sol#1301)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (UnicornMasterChef.sol#1304)
```

```
        - poolInfo[_pid].depositFeeBP = _depositFeeBP (UnicornMasterChef.sol#1305)
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (UnicornMasterChef.sol#1303)
Reentrancy in UnicornMasterChef.updateEmissionRate(uint256) (UnicornMasterChef.sol#1451-1454):
        External calls:
        - massUpdatePools() (UnicornMasterChef.sol#1452)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        State variables written after the call(s):
        - uniqPerBlock = _uniqPerBlock (UnicornMasterChef.sol#1453)
Reentrancy in UnicornMasterChef.updatePool(uint256) (UnicornMasterChef.sol#1342-1358):
        External calls:
        - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
        - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        State variables written after the call(s):
        - pool.accUNIQPerShare = pool.accUNIQPerShare.add(uniqReward.mul(1e12).div(lpSupply)) (UnicornMasterChef.sol#1356)
        - pool.lastRewardBlock = block.number (UnicornMasterChef.sol#1357)
Reentrancy in UnicornMasterChef.withdraw(uint256,uint256) (UnicornMasterChef.sol#1395-1412):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1399)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1402)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        - safeUNIQTransfer(msg.sender,pending.sub(comissionPaid)) (UnicornMasterChef.sol#1403)
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (UnicornMasterChef.sol#1408)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accUNIQPerShare).div(1e12) (UnicornMasterChef.sol#1410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
UnicornToken.addLiquidity(uint256,uint256) (UnicornMasterChef.sol#1164-1177) ignores return value by uniswapV2Router.addLiquidityETH{valu
e: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000000000dEaD),block.timestamp) (UnicornMasterChef.sol#1
169-1176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
UnicornToken.allowance(address,address).owner (UnicornMasterChef.sol#892) shadows:
        - Ownable.owner() (UnicornMasterChef.sol#286-288) (function)
UnicornToken._approve(address,address,uint256).owner (UnicornMasterChef.sol#1093) shadows:
        - Ownable.owner() (UnicornMasterChef.sol#286-288) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
UnicornToken.constructor(address,address)._projectWallet (UnicornMasterChef.sol#771) lacks a zero-check on :
                - projectWallet = _projectWallet (UnicornMasterChef.sol#775)
UnicornToken.changeProjectWallet(address).addr (UnicornMasterChef.sol#795) lacks a zero-check on :
                - projectWallet = addr (UnicornMasterChef.sol#796)
UnicornToken.setOperator(address).newOperator (UnicornMasterChef.sol#811) lacks a zero-check on :
                - operator = newOperator (UnicornMasterChef.sol#812)
UnicornMasterChef.constructor(UnicornToken,address,address,uint256,uint256)._devaddr (UnicornMasterChef.sol#1245) lacks a zero-check on :
                - devaddr = _devaddr (UnicornMasterChef.sol#1251)
UnicornMasterChef.constructor(UnicornToken,address,address,uint256,uint256)._feeAddress (UnicornMasterChef.sol#1246) lacks a zero-check o
n :
                - feeAddress = _feeAddress (UnicornMasterChef.sol#1252)
UnicornMasterChef.dev(address)._devaddr (UnicornMasterChef.sol#1437) lacks a zero-check on :
                - devaddr = _devaddr (UnicornMasterChef.sol#1439)
UnicornMasterChef.setFeeAddress(address)._feeAddress (UnicornMasterChef.sol#1445) lacks a zero-check on :
                - feeAddress = _feeAddress (UnicornMasterChef.sol#1447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in UnicornToken.constructor(address,address) (UnicornMasterChef.sol#771-787):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (UnicornMasterC
```

```
hef.sol#781-782)
        State variables written after the call(s):
        - uniswapV2Router = _uniswapV2Router (UnicornMasterChef.sol#785)
Reentrancy in UnicornToken.swapAndLiquify(uint256) (UnicornMasterChef.sol#1123-1144):
        External calls:
        - swapTokensForEth(half) (UnicornMasterChef.sol#1135)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornMasterChef.sol#1155-1161)
        - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
                - _allowances[owner][spender] = amount (UnicornMasterChef.sol#1097)
Reentrancy in UnicornToken.transferFrom(address,address,uint256) (UnicornMasterChef.sol#921-929):
        External calls:
        - _transfer(sender,recipient,amount) (UnicornMasterChef.sol#922)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornMasterChef.sol#1155-1161)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (UnicornMasterChef.sol#922)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (UnicornMa
sterChef.sol#923-927)
                - _allowances[owner][spender] = amount (UnicornMasterChef.sol#1097)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in UnicornMasterChef.deposit(uint256,uint256,address) (UnicornMasterChef.sol#1361-1392):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1364)
```

```
        - updatePool(_pid) (UnicornMasterChef.sol#1364)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - unicornReferral.recordReferral(msg.sender,_referrer) (UnicornMasterChef.sol#1367)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1372)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (UnicornMasterChef.sol#1469)
                - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1372)
Reentrancy in UnicornMasterChef.deposit(uint256,uint256,address) (UnicornMasterChef.sol#1361-1392):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1364)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - unicornReferral.recordReferral(msg.sender,_referrer) (UnicornMasterChef.sol#1367)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1372)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        - safeUNIQTransfer(msg.sender,pending.sub(comissionPaid)) (UnicornMasterChef.sol#1373)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (UnicornMasterChef.sol#1378)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (UnicornMasterChef.sol#1384)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (UnicornMasterChef.sol#1391)
Reentrancy in UnicornMasterChef.emergencyWithdraw(uint256) (UnicornMasterChef.sol#1416-1424):
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (UnicornMasterChef.sol#1422)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (UnicornMasterChef.sol#1423)
Reentrancy in UnicornMasterChef.payReferralCommission(address,uint256) (UnicornMasterChef.sol#1458-1476):
        External calls:
        - safeUNIQTransfer(referrer,commissionAmount) (UnicornMasterChef.sol#1467)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (UnicornMasterChef.sol#1469)
Reentrancy in UnicornToken.swapAndLiquify(uint256) (UnicornMasterChef.sol#1123-1144):
        External calls:
        - swapTokensForEth(half) (UnicornMasterChef.sol#1135)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornMasterChef.sol#1155-1161)
        - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (UnicornMasterChef.sol#1098)
                - addLiquidity(otherHalf,newBalance) (UnicornMasterChef.sol#1141)
        - SwapAndLiquify(half,newBalance,otherHalf) (UnicornMasterChef.sol#1143)
Reentrancy in UnicornToken.transferFrom(address,address,uint256) (UnicornMasterChef.sol#921-929):
        External calls:
        - _transfer(sender,recipient,amount) (UnicornMasterChef.sol#922)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornMasterChef.sol#1155-1161)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (UnicornMasterChef.sol#922)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
```

```
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornMasterChef.sol#1169-1176)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (UnicornMasterChef.sol#1098)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (U
nicornMasterChef.sol#923-927)
Reentrancy in UnicornMasterChef.withdraw(uint256,uint256) (UnicornMasterChef.sol#1395-1412):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1399)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1402)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (UnicornMasterChef.sol#1469)
                - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1402)
Reentrancy in UnicornMasterChef.withdraw(uint256,uint256) (UnicornMasterChef.sol#1395-1412):
        External calls:
        - updatePool(_pid) (UnicornMasterChef.sol#1399)
                - uniq.mint(devaddr,uniqReward.div(10)) (UnicornMasterChef.sol#1354)
                - uniq.mint(address(this),uniqReward) (UnicornMasterChef.sol#1355)
        - comissionPaid = payReferralCommission(msg.sender,pending) (UnicornMasterChef.sol#1402)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
                - unicornReferral.recordReferralCommission(referrer,commissionAmount) (UnicornMasterChef.sol#1468)
        - safeUNIQTransfer(msg.sender,pending.sub(comissionPaid)) (UnicornMasterChef.sol#1403)
                - uniq.transfer(_to,uniqBal) (UnicornMasterChef.sol#1430)
                - uniq.transfer(_to,_amount) (UnicornMasterChef.sol#1432)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (UnicornMasterChef.sol#1408)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (UnicornMasterChef.sol#1411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (UnicornMasterChef.sol#426-437) uses assembly
        - INLINE ASM (UnicornMasterChef.sol#433-435)
Address._functionCallWithValue(address,bytes,uint256,string) (UnicornMasterChef.sol#534-560) uses assembly
```

```
        - INLINE ASM (UnicornMasterChef.sol#552-555)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
UnicornToken._transfer(address,address,uint256) (UnicornMasterChef.sol#982-1042) compares to a boolean constant:
        -whitelist[sender] == true (UnicornMasterChef.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (UnicornMasterChef.sol#481-483) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (UnicornMasterChef.sol#510-516) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (UnicornMasterChef.sol#524-532) is never used and should be removed
Address.sendValue(address,uint256) (UnicornMasterChef.sol#455-461) is never used and should be removed
Context._msgData() (UnicornMasterChef.sol#264-267) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (UnicornMasterChef.sol#201-215) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (UnicornMasterChef.sol#226-236) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (UnicornMasterChef.sol#217-224) is never used and should be removed
SafeMath.mod(uint256,uint256) (UnicornMasterChef.sol#140-142) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (UnicornMasterChef.sol#156-159) is never used and should be removed
UnicornToken._burn(address,uint256) (UnicornMasterChef.sol#1072-1078) is never used and should be removed
UnicornToken._burnFrom(address,uint256) (UnicornMasterChef.sol#1107-1110) is never used and should be removed
UnicornToken.mint(uint256) (UnicornMasterChef.sol#975-978) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (UnicornMasterChef.sol#455-461):
        - (success) = recipient.call{value: amount}() (UnicornMasterChef.sol#459)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (UnicornMasterChef.sol#534-560):
        - (success,returndata) = target.call{value: weiValue}(data) (UnicornMasterChef.sol#543)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
INFO:Detectors:
Function IUniswapV2Router01.WETH() (UnicornMasterChef.sol#566) is not in mixedCase
Variable UnicornToken.MAX_SUPPLY (UnicornMasterChef.sol#744) is not in mixedCase
Parameter UnicornMasterChef.setReferralCommissionRate(uint16)._referralCommissionRate (UnicornMasterChef.sol#1263) is not in mixedCase
Parameter UnicornMasterChef.add(uint256,IBEP20,uint16,bool)._allocPoint (UnicornMasterChef.sol#1270) is not in mixedCase
Parameter UnicornMasterChef.add(uint256,IBEP20,uint16,bool)._lpToken (UnicornMasterChef.sol#1271) is not in mixedCase
Parameter UnicornMasterChef.add(uint256,IBEP20,uint16,bool)._depositFeeBP (UnicornMasterChef.sol#1272) is not in mixedCase
Parameter UnicornMasterChef.add(uint256,IBEP20,uint16,bool)._withUpdate (UnicornMasterChef.sol#1273) is not in mixedCase
Parameter UnicornMasterChef.set(uint256,uint256,uint16,bool)._pid (UnicornMasterChef.sol#1294) is not in mixedCase
Parameter UnicornMasterChef.set(uint256,uint256,uint16,bool)._allocPoint (UnicornMasterChef.sol#1295) is not in mixedCase
Parameter UnicornMasterChef.set(uint256,uint256,uint16,bool)._depositFeeBP (UnicornMasterChef.sol#1296) is not in mixedCase
Parameter UnicornMasterChef.set(uint256,uint256,uint16,bool)._withUpdate (UnicornMasterChef.sol#1297) is not in mixedCase
Parameter UnicornMasterChef.getMultiplier(uint256,uint256)._from (UnicornMasterChef.sol#1309) is not in mixedCase
Parameter UnicornMasterChef.getMultiplier(uint256,uint256)._to (UnicornMasterChef.sol#1309) is not in mixedCase
Parameter UnicornMasterChef.pendingUNIQ(uint256,address)._pid (UnicornMasterChef.sol#1314) is not in mixedCase
Parameter UnicornMasterChef.pendingUNIQ(uint256,address)._user (UnicornMasterChef.sol#1314) is not in mixedCase
Parameter UnicornMasterChef.setUnicornReferral(IUnicornReferral)._unicornReferral (UnicornMasterChef.sol#1329) is not in mixedCase
Parameter UnicornMasterChef.updatePool(uint256)._pid (UnicornMasterChef.sol#1342) is not in mixedCase
Parameter UnicornMasterChef.deposit(uint256,uint256,address)._pid (UnicornMasterChef.sol#1361) is not in mixedCase
Parameter UnicornMasterChef.deposit(uint256,uint256,address)._amount (UnicornMasterChef.sol#1361) is not in mixedCase
Parameter UnicornMasterChef.deposit(uint256,uint256,address)._referrer (UnicornMasterChef.sol#1361) is not in mixedCase
Parameter UnicornMasterChef.withdraw(uint256,uint256)._pid (UnicornMasterChef.sol#1395) is not in mixedCase
Parameter UnicornMasterChef.withdraw(uint256,uint256)._amount (UnicornMasterChef.sol#1395) is not in mixedCase
Parameter UnicornMasterChef.emergencyWithdraw(uint256)._pid (UnicornMasterChef.sol#1416) is not in mixedCase
Parameter UnicornMasterChef.safeUNIQTransfer(address,uint256)._to (UnicornMasterChef.sol#1427) is not in mixedCase
Parameter UnicornMasterChef.safeUNIQTransfer(address,uint256)._amount (UnicornMasterChef.sol#1427) is not in mixedCase
Parameter UnicornMasterChef.dev(address)._devaddr (UnicornMasterChef.sol#1437) is not in mixedCase
Parameter UnicornMasterChef.setFeeAddress(address)._feeAddress (UnicornMasterChef.sol#1445) is not in mixedCase
Parameter UnicornMasterChef.updateEmissionRate(uint256)._uniqPerBlock (UnicornMasterChef.sol#1451) is not in mixedCase
Parameter UnicornMasterChef.payReferralCommission(address,uint256)._user (UnicornMasterChef.sol#1458) is not in mixedCase
Parameter UnicornMasterChef.payReferralCommission(address,uint256)._pending (UnicornMasterChef.sol#1458) is not in mixedCase
Variable UnicornMasterChef.MAX_FEE (UnicornMasterChef.sol#1227) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (UnicornMasterChef.sol#265)" inContext (UnicornMasterChef.sol#259-268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
```

```
INFO:Detectors:
UnicornMasterChef.MAX_FEE (UnicornMasterChef.sol#1227) should be constant
UnicornToken.MAX_SUPPLY (UnicornMasterChef.sol#744) should be constant
UnicornToken.lpTax (UnicornMasterChef.sol#734) should be constant
UnicornToken.projectTax (UnicornMasterChef.sol#733) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (UnicornMasterChef.sol#286-288)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (UnicornMasterChef.sol#305-308)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (UnicornMasterChef.sol#314-318)
changeProjectWallet(address) should be declared external:
        - UnicornToken.changeProjectWallet(address) (UnicornMasterChef.sol#795-797)
setNumTokensSellToAddToLiquidity(uint256) should be declared external:
        - UnicornToken.setNumTokensSellToAddToLiquidity(uint256) (UnicornMasterChef.sol#800-802)
addOrRemoveFromWhitelist(address,bool) should be declared external:
        - UnicornToken.addOrRemoveFromWhitelist(address,bool) (UnicornMasterChef.sol#806-808)
setOperator(address) should be declared external:
        - UnicornToken.setOperator(address) (UnicornMasterChef.sol#811-813)
enableDisableFee(bool) should be declared external:
        - UnicornToken.enableDisableFee(bool) (UnicornMasterChef.sol#817-819)
setLiquification(bool) should be declared external:
        - UnicornToken.setLiquification(bool) (UnicornMasterChef.sol#823-825)
mint(address,uint256) should be declared external:
        - UnicornToken.mint(address,uint256) (UnicornMasterChef.sol#830-833)
name() should be declared external:
        - UnicornToken.name() (UnicornMasterChef.sol#839-841)
symbol() should be declared external:
        - UnicornToken.symbol() (UnicornMasterChef.sol#847-849)
decimals() should be declared external:
        - UnicornToken.decimals() (UnicornMasterChef.sol#856-858)
transfer(address,uint256) should be declared external:
        - UnicornToken.transfer(address,uint256) (UnicornMasterChef.sol#882-887)
allowance(address,address) should be declared external:
```

```
transfer(address,uint256) should be declared external:
        - UnicornToken.transfer(address,uint256) (UnicornMasterChef.sol#882-887)
allowance(address,address) should be declared external:
        - UnicornToken.allowance(address,address) (UnicornMasterChef.sol#892-894)
approve(address,uint256) should be declared external:
        - UnicornToken.approve(address,uint256) (UnicornMasterChef.sol#903-907)
transferFrom(address,address,uint256) should be declared external:
        - UnicornToken.transferFrom(address,address,uint256) (UnicornMasterChef.sol#921-929)
increaseAllowance(address,uint256) should be declared external:
        - UnicornToken.increaseAllowance(address,uint256) (UnicornMasterChef.sol#943-946)
decreaseAllowance(address,uint256) should be declared external:
        - UnicornToken.decreaseAllowance(address,uint256) (UnicornMasterChef.sol#962-965)
setReferralCommissionRate(uint16) should be declared external:
        - UnicornMasterChef.setReferralCommissionRate(uint16) (UnicornMasterChef.sol#1263-1266)
add(uint256,IBEP20,uint16,bool) should be declared external:
        - UnicornMasterChef.add(uint256,IBEP20,uint16,bool) (UnicornMasterChef.sol#1269-1290)
set(uint256,uint256,uint16,bool) should be declared external:
        - UnicornMasterChef.set(uint256,uint256,uint16,bool) (UnicornMasterChef.sol#1293-1306)
setUnicornReferral(IUnicornReferral) should be declared external:
        - UnicornMasterChef.setUnicornReferral(IUnicornReferral) (UnicornMasterChef.sol#1329-1331)
withdraw(uint256,uint256) should be declared external:
        - UnicornMasterChef.withdraw(uint256,uint256) (UnicornMasterChef.sol#1395-1412)
emergencyWithdraw(uint256) should be declared external:
        - UnicornMasterChef.emergencyWithdraw(uint256) (UnicornMasterChef.sol#1416-1424)
dev(address) should be declared external:
        - UnicornMasterChef.dev(address) (UnicornMasterChef.sol#1437-1440)
setFeeAddress(address) should be declared external:
        - UnicornMasterChef.setFeeAddress(address) (UnicornMasterChef.sol#1445-1448)
updateEmissionRate(uint256) should be declared external:
        - UnicornMasterChef.updateEmissionRate(uint256) (UnicornMasterChef.sol#1451-1454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UnicornMasterChef.sol analyzed (12 contracts with 75 detectors), 120 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Slither log >> UnicornPresale.sol

```
INFO:Detectors:
UnicornPresale.safeTransferTokens(uint256) (UnicornPresale.sol#225-230) ignores return value by token.transfer(msg.sender,noOfTokens) (Un
icornPresale.sol#228)
UnicornPresale.withdrawTokens() (UnicornPresale.sol#289-293) ignores return value by token.transfer(owner,getTokenBalance()) (UnicornPres
ale.sol#291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
UnicornPresale.changeOwner(address) (UnicornPresale.sol#205-209) should emit an event for:
        - owner = addr (UnicornPresale.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
UnicornPresale.setPresaleStarts(uint256) (UnicornPresale.sol#193-195) should emit an event for:
        - presaleStarts = time (UnicornPresale.sol#194)
UnicornPresale.changePrice(uint256) (UnicornPresale.sol#210-213) should emit an event for:
        - _price = price (UnicornPresale.sol#212)
UnicornPresale.setMaxLimit(uint256) (UnicornPresale.sol#236-240) should emit an event for:
        - MAX_BUY_LIMIT = maxL (UnicornPresale.sol#239)
UnicornPresale.changePresaleEndTime(uint256) (UnicornPresale.sol#306-310) should emit an event for:
        - presaleTimeEnds = time (UnicornPresale.sol#308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
UnicornPresale.changeOwner(address).addr (UnicornPresale.sol#205) lacks a zero-check on :
                - owner = addr (UnicornPresale.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in UnicornPresale.safeTransferTokens(uint256) (UnicornPresale.sol#225-230):
        External calls:
        - token.transfer(msg.sender,noOfTokens) (UnicornPresale.sol#228)
        State variables written after the call(s):
        - totalTokenSold = totalTokenSold.add(noOfTokens) (UnicornPresale.sol#229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
UnicornPresale.buyToken() (UnicornPresale.sol#242-257) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(presaleTimeEnds > block.timestamp,Presale Finished) (UnicornPresale.sol#243)
        - require(bool,string)(presaleStarts < block.timestamp,Presale not started) (UnicornPresale.sol#245)
```

```
        - require(bool,string)(presaleStarts < block.timestamp,Presale not started) (UnicornPresale.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SafeMath.mod(uint256,uint256) (UnicornPresale.sol#129-131) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (UnicornPresale.sol#144-147) is never used and should be removed
SafeMath.sub(uint256,uint256) (UnicornPresale.sol#40-42) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (UnicornPresale.sol#53-58) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable UnicornPresale._price (UnicornPresale.sol#160) is not in mixedCase
Variable UnicornPresale.MAX_BUY_LIMIT (UnicornPresale.sol#166) is not in mixedCase
Modifier UnicornPresale.IsOwner() (UnicornPresale.sol#197-200) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
UnicornPresale.normalSaleSold (UnicornPresale.sol#173) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
setPresaleStarts(uint256) should be declared external:
        - UnicornPresale.setPresaleStarts(uint256) (UnicornPresale.sol#193-195)
changeOwner(address) should be declared external:
        - UnicornPresale.changeOwner(address) (UnicornPresale.sol#205-209)
changePrice(uint256) should be declared external:
        - UnicornPresale.changePrice(uint256) (UnicornPresale.sol#210-213)
setMaxLimit(uint256) should be declared external:
        - UnicornPresale.setMaxLimit(uint256) (UnicornPresale.sol#236-240)
buyToken() should be declared external:
        - UnicornPresale.buyToken() (UnicornPresale.sol#242-257)
withdrawTokens() should be declared external:
        - UnicornPresale.withdrawTokens() (UnicornPresale.sol#289-293)
withdrawBalance() should be declared external:
        - UnicornPresale.withdrawBalance() (UnicornPresale.sol#298-301)
changePresaleEndTime(uint256) should be declared external:
        - UnicornPresale.changePresaleEndTime(uint256) (UnicornPresale.sol#306-310)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UnicornPresale.sol analyzed (3 contracts with 75 detectors), 26 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> UnicornReferral.sol

```
INFO:Detectors:
Address.isContract(address) (UnicornReferral.sol#145-154) uses assembly
        - INLINE ASM (UnicornReferral.sol#152)
Address._verifyCallResult(bool,bytes,string) (UnicornReferral.sol#290-307) uses assembly
        - INLINE ASM (UnicornReferral.sol#299-302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.12', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
        - >=0.6.0<0.8.0 (UnicornReferral.sol#9)
        - >=0.6.0<0.8.0 (UnicornReferral.sol#34)
        - 0.6.12 (UnicornReferral.sol#101)
        - >=0.6.2<0.8.0 (UnicornReferral.sol#122)
        - >=0.6.0<0.8.0 (UnicornReferral.sol#312)
        - ^0.6.0 (UnicornReferral.sol#527)
        - >=0.4.0 (UnicornReferral.sol#628)
        - 0.6.12 (UnicornReferral.sol#727)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (UnicornReferral.sol#290-307) is never used and should be removed
Address.functionCall(address,bytes) (UnicornReferral.sol#198-200) is never used and should be removed
Address.functionCall(address,bytes,string) (UnicornReferral.sol#208-210) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (UnicornReferral.sol#223-225) is never used and should be removed
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
Address.functionCallWithValue(address,bytes,uint256) (UnicornReferral.sol#223-225) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (UnicornReferral.sol#233-240) is never used and should be removed
Address.functionDelegateCall(address,bytes) (UnicornReferral.sol#272-274) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (UnicornReferral.sol#282-288) is never used and should be removed
Address.functionStaticCall(address,bytes) (UnicornReferral.sol#248-250) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (UnicornReferral.sol#258-264) is never used and should be removed
Address.isContract(address) (UnicornReferral.sol#145-154) is never used and should be removed
Address.sendValue(address,uint256) (UnicornReferral.sol#172-178) is never used and should be removed
Context._msgData() (UnicornReferral.sol#26-29) is never used and should be removed
SafeBEP20._callOptionalReturn(IBEP20,bytes) (UnicornReferral.sol#612-623) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (UnicornReferral.sol#569-583) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (UnicornReferral.sol#594-604) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (UnicornReferral.sol#585-592) is never used and should be removed
SafeBEP20.safeTransfer(IBEP20,address,uint256) (UnicornReferral.sol#545-551) is never used and should be removed
SafeBEP20.safeTransferFrom(IBEP20,address,address,uint256) (UnicornReferral.sol#553-560) is never used and should be removed
SafeMath.add(uint256,uint256) (UnicornReferral.sol#394-398) is never used and should be removed
SafeMath.div(uint256,uint256) (UnicornReferral.sol#444-447) is never used and should be removed
SafeMath.div(uint256,uint256,string) (UnicornReferral.sol#499-502) is never used and should be removed
SafeMath.mod(uint256,uint256) (UnicornReferral.sol#461-464) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (UnicornReferral.sol#519-522) is never used and should be removed
SafeMath.mul(uint256,uint256) (UnicornReferral.sol#425-430) is never used and should be removed
SafeMath.sub(uint256,uint256) (UnicornReferral.sol#410-413) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (UnicornReferral.sol#479-482) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (UnicornReferral.sol#333-337) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (UnicornReferral.sol#369-372) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (UnicornReferral.sol#379-382) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (UnicornReferral.sol#354-362) is never used and should be removed
SafeMath.trySub(uint256,uint256) (UnicornReferral.sol#344-347) 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 (UnicornReferral.sol#9) is too complex
Pragma version>=0.6.0<0.8.0 (UnicornReferral.sol#34) is too complex
Pragma version>=0.6.2<0.8.0 (UnicornReferral.sol#122) is too complex
Pragma version>=0.6.0<0.8.0 (UnicornReferral.sol#312) is too complex
Pragma version^0.6.0 (UnicornReferral.sol#527) allows old versions
Pragma version>=0.4.0 (UnicornReferral.sol#628) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (UnicornReferral.sol#172-178):
        - (success) = recipient.call{value: amount}() (UnicornReferral.sol#176)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (UnicornReferral.sol#233-240):
        - (success,returndata) = target.call{value: value}(data) (UnicornReferral.sol#238)
Low level call in Address.functionStaticCall(address,bytes,string) (UnicornReferral.sol#258-264):
        - (success,returndata) = target.staticcall(data) (UnicornReferral.sol#262)
Low level call in Address.functionDelegateCall(address,bytes,string) (UnicornReferral.sol#282-288):
        - (success,returndata) = target.delegatecall(data) (UnicornReferral.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter UnicornReferral.recordReferral(address,address)._user (UnicornReferral.sol#750) is not in mixedCase
Parameter UnicornReferral.recordReferral(address,address)._referrer (UnicornReferral.sol#750) is not in mixedCase
Parameter UnicornReferral.recordReferralCommission(address,uint256)._referrer (UnicornReferral.sol#762) is not in mixedCase
Parameter UnicornReferral.recordReferralCommission(address,uint256)._commission (UnicornReferral.sol#762) is not in mixedCase
Parameter UnicornReferral.getReferrer(address)._user (UnicornReferral.sol#770) is not in mixedCase
Parameter UnicornReferral.updateOperator(address,bool)._operator (UnicornReferral.sol#775) is not in mixedCase
Parameter UnicornReferral.updateOperator(address,bool)._status (UnicornReferral.sol#775) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (UnicornReferral.sol#27)" inContext (UnicornReferral.sol#21-30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (UnicornReferral.sol#84-87)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (UnicornReferral.sol#93-97)
recordReferral(address,address) should be declared external:
        - UnicornReferral.recordReferral(address,address) (UnicornReferral.sol#750-760)
recordReferralCommission(address,uint256) should be declared external:
        - UnicornReferral.recordReferralCommission(address,uint256) (UnicornReferral.sol#762-767)
getReferrer(address) should be declared external:
        - UnicornReferral.getReferrer(address) (UnicornReferral.sol#770-772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UnicornReferral.sol analyzed (8 contracts with 75 detectors), 57 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> UnicornToken.sol

```
INFO:Detectors:
UnicornToken.addLiquidity(uint256,uint256) (UnicornToken.sol#1016-1029) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000000000000d
EaD),block.timestamp) (UnicornToken.sol#1021-1028)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
UnicornToken.addLiquidity(uint256,uint256) (UnicornToken.sol#1016-1029) ignores return value by uniswapV2Router.addLiquidityETH{value: et
hAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
UnicornToken.allowance(address,address).owner (UnicornToken.sol#744) shadows:
        - Ownable.owner() (UnicornToken.sol#62-64) (function)
UnicornToken._approve(address,address,uint256).owner (UnicornToken.sol#945) shadows:
        - Ownable.owner() (UnicornToken.sol#62-64) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
UnicornToken.constructor(address,address)._projectWallet (UnicornToken.sol#623) lacks a zero-check on :
                - projectWallet = _projectWallet (UnicornToken.sol#627)
UnicornToken.changeProjectWallet(address).addr (UnicornToken.sol#647) lacks a zero-check on :
                - projectWallet = addr (UnicornToken.sol#648)
UnicornToken.setOperator(address).newOperator (UnicornToken.sol#663) lacks a zero-check on :
                - operator = newOperator (UnicornToken.sol#664)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in UnicornToken.constructor(address,address) (UnicornToken.sol#623-639):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (UnicornToken.s
ol#633-634)
        State variables written after the call(s):
        - uniswapV2Router = _uniswapV2Router (UnicornToken.sol#637)
Reentrancy in UnicornToken.swapAndLiquify(uint256) (UnicornToken.sol#975-996):
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
        External calls:
        - swapTokensForEth(half) (UnicornToken.sol#987)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornToken.sol#1007-1013)
        - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
                - _allowances[owner][spender] = amount (UnicornToken.sol#949)
Reentrancy in UnicornToken.transferFrom(address,address,uint256) (UnicornToken.sol#773-781):
        External calls:
        - _transfer(sender,recipient,amount) (UnicornToken.sol#774)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornToken.sol#1007-1013)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (UnicornToken.sol#774)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (UnicornTo
ken.sol#775-779)
                - _allowances[owner][spender] = amount (UnicornToken.sol#949)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in UnicornToken.swapAndLiquify(uint256) (UnicornToken.sol#975-996):
        External calls:
        - swapTokensForEth(half) (UnicornToken.sol#987)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornToken.sol#1007-1013)
        - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
```

```
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (UnicornToken.sol#950)
                - addLiquidity(otherHalf,newBalance) (UnicornToken.sol#993)
        - SwapAndLiquify(half,newBalance,otherHalf) (UnicornToken.sol#995)
Reentrancy in UnicornToken.transferFrom(address,address,uint256) (UnicornToken.sol#773-781):
        External calls:
        - _transfer(sender,recipient,amount) (UnicornToken.sol#774)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (U
nicornToken.sol#1007-1013)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (UnicornToken.sol#774)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000
0000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (UnicornToken.sol#950)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (U
nicornToken.sol#775-779)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
UnicornToken._transfer(address,address,uint256) (UnicornToken.sol#834-894) compares to a boolean constant:
        -whitelist[sender] == true (UnicornToken.sol#842)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.6.0<0.8.0', '>=0.6.4', '^0.6.12']
        - >=0.6.0<0.8.0 (UnicornToken.sol#7)
        - >=0.6.0<0.8.0 (UnicornToken.sol#31)
        - >=0.6.0<0.8.0 (UnicornToken.sol#97)
        - >=0.6.4 (UnicornToken.sol#257)
        - ^0.6.12 (UnicornToken.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
INFO:Detectors:
Context._msgData() (UnicornToken.sol#24-27) is never used and should be removed
SafeMath.mod(uint256,uint256) (UnicornToken.sol#233-235) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (UnicornToken.sol#249-252) is never used and should be removed
UnicornToken._burn(address,uint256) (UnicornToken.sol#924-930) is never used and should be removed
UnicornToken._burnFrom(address,uint256) (UnicornToken.sol#959-962) is never used and should be removed
UnicornToken.mint(uint256) (UnicornToken.sol#827-830) 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 (UnicornToken.sol#7) is too complex
Pragma version>=0.6.0<0.8.0 (UnicornToken.sol#31) is too complex
Pragma version>=0.6.0<0.8.0 (UnicornToken.sol#97) is too complex
Pragma version>=0.6.4 (UnicornToken.sol#257) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (UnicornToken.sol#383) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (UnicornToken.sol#384) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (UnicornToken.sol#401) is not in mixedCase
Function IUniswapV2Router01.WETH() (UnicornToken.sol#423) is not in mixedCase
Variable UnicornToken.MAX_SUPPLY (UnicornToken.sol#596) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (UnicornToken.sol#25)" inContext (UnicornToken.sol#19-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (UnicornToken.so
l#428) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
(UnicornToken.sol#429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
INFO:Detectors:
UnicornToken.addLiquidity(uint256,uint256) (UnicornToken.sol#1016-1029) uses literals with too many digits:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0x000000000000000000000000000000000000dEaD),block.timestamp) (UnicornToken.sol#1021-1028)
UnicornToken.slitherConstructorVariables() (UnicornToken.sol#571-1038) uses literals with too many digits:
        - MAX_SUPPLY = 500000 * 1e18 (UnicornToken.sol#596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
UnicornToken.MAX_SUPPLY (UnicornToken.sol#596) should be constant
UnicornToken.lpTax (UnicornToken.sol#586) should be constant
UnicornToken.projectTax (UnicornToken.sol#585) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (UnicornToken.sol#62-64)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (UnicornToken.sol#81-84)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (UnicornToken.sol#90-94)
changeProjectWallet(address) should be declared external:
        - UnicornToken.changeProjectWallet(address) (UnicornToken.sol#647-649)
setNumTokensSellToAddToLiquidity(uint256) should be declared external:
        - UnicornToken.setNumTokensSellToAddToLiquidity(uint256) (UnicornToken.sol#652-654)
addOrRemoveFromWhitelist(address,bool) should be declared external:
        - UnicornToken.addOrRemoveFromWhitelist(address,bool) (UnicornToken.sol#658-660)
setOperator(address) should be declared external:
        - UnicornToken.setOperator(address) (UnicornToken.sol#663-665)
enableDisableFee(bool) should be declared external:
        - UnicornToken.enableDisableFee(bool) (UnicornToken.sol#669-671)
setLiquification(bool) should be declared external:
        - UnicornToken.setLiquification(bool) (UnicornToken.sol#675-677)
mint(address,uint256) should be declared external:
        - UnicornToken.mint(address,uint256) (UnicornToken.sol#682-685)
name() should be declared external:
        - UnicornToken.name() (UnicornToken.sol#691-693)
symbol() should be declared external:
        - UnicornToken.symbol() (UnicornToken.sol#699-701)
decimals() should be declared external:
```

```
decimals() should be declared external:
        - UnicornToken.decimals() (UnicornToken.sol#708-710)
transfer(address,uint256) should be declared external:
        - UnicornToken.transfer(address,uint256) (UnicornToken.sol#734-739)
allowance(address,address) should be declared external:
        - UnicornToken.allowance(address,address) (UnicornToken.sol#744-746)
approve(address,uint256) should be declared external:
        - UnicornToken.approve(address,uint256) (UnicornToken.sol#755-759)
transferFrom(address,address,uint256) should be declared external:
        - UnicornToken.transferFrom(address,address,uint256) (UnicornToken.sol#773-781)
increaseAllowance(address,uint256) should be declared external:
        - UnicornToken.increaseAllowance(address,uint256) (UnicornToken.sol#795-798)
decreaseAllowance(address,uint256) should be declared external:
        - UnicornToken.decreaseAllowance(address,uint256) (UnicornToken.sol#814-817)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UnicornToken.sol analyzed (9 contracts with 75 detectors), 55 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Timelock.sol

```
INFO:Detectors:
Timelock.constructor(address,uint256).admin_ (Timelock.sol#163) lacks a zero-check on :
                - admin = admin_ (Timelock.sol#167)
Timelock.setPendingAdmin(address).pendingAdmin_ (Timelock.sol#192) lacks a zero-check on :
                - pendingAdmin = pendingAdmin_ (Timelock.sol#200)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (Timelock.sol#225) lacks a zero-check on :
                - (success,returnData) = target.call.value(value)(callData) (Timelock.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#225-250):
        External calls:
        - (success,returnData) = target.call.value(value)(callData) (Timelock.sol#244)
        Event emitted after the call(s):
        - ExecuteTransaction(txHash,target,value,signature,data,eta) (Timelock.sol#247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#205-214) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(eta >= getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block must satisfy delay.) (Timelock.sol#207)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#225-250) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction: Transaction hasn't surpassed time lock.) (Timelock.sol#230)
        - require(bool,string)(getBlockTimestamp() <= eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (Timelock.sol#231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SafeMath.div(uint256,uint256) (Timelock.sol#84-86) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Timelock.sol#99-106) is never used and should be removed
SafeMath.mod(uint256,uint256) (Timelock.sol#119-121) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Timelock.sol#134-137) is never used and should be removed
SafeMath.mul(uint256,uint256) (Timelock.sol#59-71) is never used and should be removed
SafeMath.sub(uint256,uint256) (Timelock.sol#30-32) is never used and should be removed
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
SafeMath.sub(uint256,uint256,string) (Timelock.sol#43-48) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#225-250):
        - (success,returnData) = target.call.value(value)(callData) (Timelock.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable Timelock.admin_initialized (Timelock.sol#158) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
setDelay(uint256) should be declared external:
        - Timelock.setDelay(uint256) (Timelock.sol#175-182)
acceptAdmin() should be declared external:
        - Timelock.acceptAdmin() (Timelock.sol#184-190)
setPendingAdmin(address) should be declared external:
        - Timelock.setPendingAdmin(address) (Timelock.sol#192-203)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#205-214)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#216-223)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#225-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Timelock.sol analyzed (2 contracts with 75 detectors), 21 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Solidity static analysis

CswapUniSWAP.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

UnicornMasterChef.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

UnicornPresale.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

UnicornReferral.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

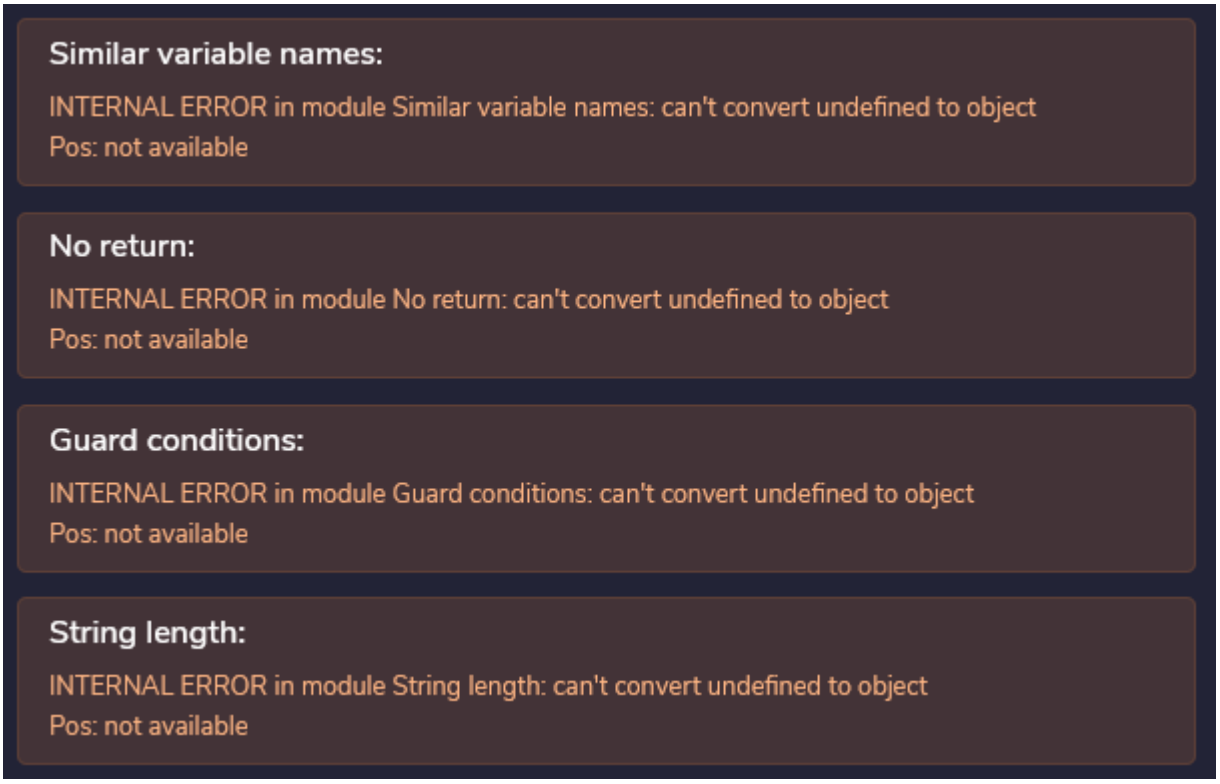
INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io



UnicornToken.sol

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Gas & Economy

Gas costs:

Gas requirement of function UnicornToken.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 90:4:

Gas costs:

Gas requirement of function UnicornToken.uniswapV2Router is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 602:4:

Gas costs:

Gas requirement of function UnicornToken.uniswapV2Pair is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 603:4:

Gas costs:

Gas requirement of function UnicornToken.changeProjectWallet is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 647:4:

Gas costs:

Gas requirement of function UnicornToken.setNumTokensSellToAddToLiquidity is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 652:4:

Gas costs:

Gas requirement of function UnicornToken.addOrRemoveFromWhitelist is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 658:4:

Gas costs:

Gas requirement of function UnicornToken.setOperator is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 663:4:

Gas costs:

Gas requirement of function UnicornToken.enableDisableFee is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 669:5:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function UnicornToken.setLiquification is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 675:4:

Gas costs:

Gas requirement of function UnicornToken.mint is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 682:4:

Gas costs:

Gas requirement of function UnicornToken.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 691:4:

Gas costs:

Gas requirement of function UnicornToken.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 699:4:

Gas costs:

Gas requirement of function UnicornToken.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 734:4:

Gas costs:

Gas requirement of function UnicornToken.approve is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 755:4:

Gas costs:

Gas requirement of function UnicornToken.transferFrom is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 773:4:

Gas costs:

Gas requirement of function UnicornToken.increaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 795:4:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function UnicornToken.decreaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 814:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 268:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 374:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 140:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 197:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 233:4:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 906:16:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 908:40:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 909:18:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 909:39:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 909:52:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 910:34:

Similar variable names:

UnicornToken._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 910:43:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 925:16:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 927:18:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 927:39:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 927:52:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 928:40:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 929:22:

Similar variable names:

UnicornToken._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 929:43:

Similar variable names:

UnicornToken._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 960:14:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Similar variable names:

UnicornToken._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 960:23:

Similar variable names:

UnicornToken._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 961:17:

Similar variable names:

UnicornToken._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 961:52:

Similar variable names:

UnicornToken._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 961:79:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 70:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 91:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 125:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 155:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 180:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 214:8:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 250:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 617:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 683:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 835:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 836:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 906:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 925:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 946:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 947:8:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 180:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 215:20:

Timelock.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Solhint Linter

CswapUniSWAP.sol

```
CswapUniSWAP.sol:3:1: Error: Compiler version ^0.6.12 does not satisfy
the r semver requirement
CswapUniSWAP.sol:55:50: Error: Use double quotes for string literals
CswapUniSWAP.sol:58:58: Error: Use double quotes for string literals
CswapUniSWAP.sol:59:26: Error: Use double quotes for string literals
CswapUniSWAP.sol:81:43: Error: Use double quotes for string literals
CswapUniSWAP.sol:114:59: Error: Use double quotes for string literals
CswapUniSWAP.sol:129:49: Error: Use double quotes for string literals
CswapUniSWAP.sol:139:37: Error: Use double quotes for string literals
CswapUniSWAP.sol:350:13: Error: Use double quotes for string literals
CswapUniSWAP.sol:371:13: Error: Use double quotes for string literals
CswapUniSWAP.sol:387:69: Error: Use double quotes for string literals
CswapUniSWAP.sol:391:53: Error: Use double quotes for string literals
```

UnicornMasterChef.sol

```
UnicornMasterChef.sol:1:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
UnicornMasterChef.sol:197:13: Error: Use double quotes for string
literals
UnicornMasterChef.sol:218:13: Error: Use double quotes for string
literals
UnicornMasterChef.sol:234:69: Error: Use double quotes for string
literals
UnicornMasterChef.sol:238:53: Error: Use double quotes for string
literals
UnicornMasterChef.sol:441:50: Error: Use double quotes for string
literals
UnicornMasterChef.sol:444:58: Error: Use double quotes for string
literals
UnicornMasterChef.sol:445:26: Error: Use double quotes for string
literals
UnicornMasterChef.sol:467:43: Error: Use double quotes for string
literals
UnicornMasterChef.sol:500:59: Error: Use double quotes for string
literals
UnicornMasterChef.sol:515:49: Error: Use double quotes for string
literals
UnicornMasterChef.sol:525:37: Error: Use double quotes for string
literals
UnicornMasterChef.sol:592:20: Error: Variable name must be in mixedCase
UnicornMasterChef.sol:638:36: Error: Code contains empty blocks
UnicornMasterChef.sol:774:59: Error: Use double quotes for string
literals
UnicornMasterChef.sol:811:97: Error: Use double quotes for string
literals
UnicornMasterChef.sol:831:39: Error: Use double quotes for string
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
literals
UnicornMasterChef.sol:832:42: Error: Use double quotes for string
literals
UnicornMasterChef.sol:839:67: Error: Use double quotes for string
literals
UnicornMasterChef.sol:845:67: Error: Use double quotes for string
literals
UnicornMasterChef.sol:869:63: Error: Use double quotes for string
literals
UnicornMasterChef.sol:902:40: Error: Use double quotes for string
literals
UnicornMasterChef.sol:921:40: Error: Use double quotes for string
literals
UnicornMasterChef.sol:923:61: Error: Use double quotes for string
literals
UnicornMasterChef.sol:942:38: Error: Use double quotes for string
literals
UnicornMasterChef.sol:943:40: Error: Use double quotes for string
literals
UnicornMasterChef.sol:957:88: Error: Use double quotes for string
literals
UnicornMasterChef.sol:1008:13: Error: Avoid to make time-based
decisions in your business logic
UnicornMasterChef.sol:1023:13: Error: Avoid to make time-based
decisions in your business logic
UnicornMasterChef.sol:1075:21: Error: Variable name must be in
mixedCase
UnicornMasterChef.sol:1123:43: Error: Use double quotes for string
literals
UnicornMasterChef.sol:1147:43: Error: Use double quotes for string
literals
UnicornMasterChef.sol:1246:41: Error: Use double quotes for string
literals
UnicornMasterChef.sol:1286:40: Error: Use double quotes for string
literals
UnicornMasterChef.sol:1294:43: Error: Use double quotes for string
literals
```

UnicornPresale.sol

```
UnicornPresale.sol:12:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
UnicornPresale.sol:166:20: Error: Variable name must be in mixedCase
UnicornPresale.sol:229:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
UnicornPresale.sol:243:35: Error: Avoid to make time-based decisions in
your business logic
UnicornPresale.sol:245:33: Error: Avoid to make time-based decisions in
your business logic
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

UnicornReferral.sol

```
UnicornReferral.sol:9:1: Error: Compiler version >=0.6.0 <0.8.0 does
not satisfy the r semver requirement
UnicornReferral.sol:34:1: Error: Compiler version >=0.6.0 <0.8.0 does
not satisfy the r semver requirement
UnicornReferral.sol:101:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
UnicornReferral.sol:122:1: Error: Compiler version >=0.6.2 <0.8.0 does
not satisfy the r semver requirement
UnicornReferral.sol:312:1: Error: Compiler version >=0.6.0 <0.8.0 does
not satisfy the r semver requirement
UnicornReferral.sol:527:1: Error: Compiler version ^0.6.0 does not
satisfy the r semver requirement
UnicornReferral.sol:628:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
UnicornReferral.sol:727:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
```

UnicornToken.sol

```
UnicornToken.sol:7:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
UnicornToken.sol:31:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
UnicornToken.sol:97:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
UnicornToken.sol:257:1: Error: Compiler version >=0.6.4 does not
satisfy the r semver requirement
UnicornToken.sol:383:5: Error: Function name must be in mixedCase
UnicornToken.sol:384:5: Error: Function name must be in mixedCase
UnicornToken.sol:401:5: Error: Function name must be in mixedCase
UnicornToken.sol:423:5: Error: Function name must be in mixedCase
UnicornToken.sol:567:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
UnicornToken.sol:596:20: Error: Variable name must be in mixedCase
UnicornToken.sol:642:36: Error: Code contains empty blocks
UnicornToken.sol:778:59: Error: Use double quotes for string literals
UnicornToken.sol:815:97: Error: Use double quotes for string literals
UnicornToken.sol:835:39: Error: Use double quotes for string literals
UnicornToken.sol:836:42: Error: Use double quotes for string literals
UnicornToken.sol:843:67: Error: Use double quotes for string literals
UnicornToken.sol:849:67: Error: Use double quotes for string literals
UnicornToken.sol:873:63: Error: Use double quotes for string literals
UnicornToken.sol:906:40: Error: Use double quotes for string literals
UnicornToken.sol:925:40: Error: Use double quotes for string literals
UnicornToken.sol:927:61: Error: Use double quotes for string literals
UnicornToken.sol:946:38: Error: Use double quotes for string literals
UnicornToken.sol:947:40: Error: Use double quotes for string literals
UnicornToken.sol:961:88: Error: Use double quotes for string literals
UnicornToken.sol:1012:13: Error: Avoid to make time-based decisions in
your business logic
UnicornToken.sol:1027:13: Error: Avoid to make time-based decisions in
your business logic
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Timelock.sol

```
Timelock.sol:2:1: Error: Compiler version 0.6.12 does not satisfy the r
semver requirement
Timelock.sol:158:17: Error: Variable name must be in mixedCase
Timelock.sol:173:32: Error: Code contains empty blocks
Timelock.sol:244:51: Error: Avoid to use ".call.value()()"
Timelock.sol:244:51: Error: Avoid to use low level calls.
Timelock.sol:254:16: Error: Avoid to make time-based decisions in your
business logic
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io