Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	MagicLamp
Website:	https://magiclamp.gold
Platform:	Binance Smart Chain
Language:	Solidity
Date:	June 30th, 2021

Table of contents

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.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the MagicLamp team to perform the Security audit of the MagicLamp protocol 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 June 30th, 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

MagicLamp is a project that aims to revolutionize the NFT market by combining all ERC20 and NFT properties on the BSC network. Users are able to buy an NFT and receive GNI accumulation & ALDN tokens proportionate to 10% of the total supply after the sale. Users also have the opportunity to stake their NFTs into MagicLamp's designed NFT Staking pool and receive even more tokens as a reward.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for MagicLamp protocol Smart Contracts
Platform	**BSC / Solidity**
File 1	MagicLamps.sol
File 1 MD5 Hash	2E15B0FBF1639B480D7171702903D8D1
File 2	GenieToken.sol
File 2 MD5 Hash	DBD69FD971E5151D906F36965D02FCAE
File 3	MagicLampWallet.sol
File 3 MD5 Hash	47D06E107B2B82C56CDE2A5B2122AFB2
File 4	ALDN.sol
File 4 MD5 Hash	CCFC2DAB61EB9475AF1B9CE701BF694D
File 5	LPStaking.sol
File 5 MD5 Hash	4A06FBCBD3126D4D486AD9B90DB1C94B
File 6	NFTStaking.sol
File 6 MD5 Hash	A899601FAD33C964C3ED9AD2D75D5204
File 7	SwapAndLiquify.sol
File 7 MD5 Hash	EFA60DEF3847B259B0D84836AD2D1111
Audit Date	June 30th, 2021
Revision Date	July 7th, 2021

Updated Zip file MD5 Hash: 14FD6D27F15DCF14BF497C81817A675E

PS: There are some external libraries and other contracts inherited, which are not included in the audit scope and thus are not audited.

Claimed Smart Contract Features

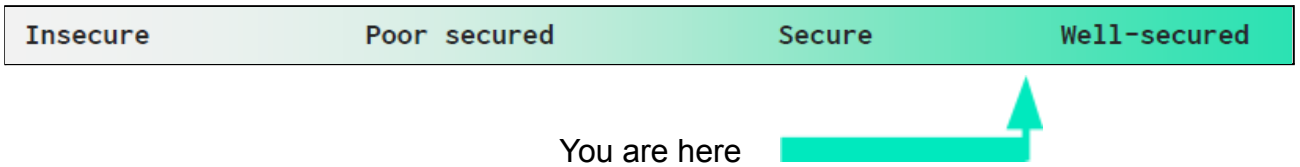
Claimed Feature Detail	Our Observation
File 1: MagicLamps.sol • Referral Reward Percentage: 10% • Prize Fund Percentage: 10% • Liquidity Fund Percentage: 10% • distributeReferralRewards: The Owner can check started and ended sale time and distribute referral rewards amounts.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 2: GenieToken.sol • Genie token is used to change MagicLamp NFT. • Number of GNIs emitted per year per NFT: 3,660 • Number of GNIs required for one name change: 1,830 • Bonus GNIs for initial sales phase: 1,830	**YES, This is valid.**
File 3: MagicLampWallet.sol • MagicLampWallet owners can withdraw, send, swap: ERC20, ERC721,Erc1155 tokens From magicLamp Wallet.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 4: ALDN.sol • Name: Magiclamp Governance Token	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of**

• Symbol: ALDN • Decimals: 9 • TaxFee: 5% • LiquidityFee: 5% • ALDN owners can set tax fee percentage, LiquidityFeePercent, MaxTxPercent, swapAndLiquifyAddress,SwapAndLiquify enable status, etc. • Also check excluded accounts and included accounts exist or not.	**the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 5: LPStaking.sol • LPSTaking With PancakeSwap Liquidity, users can stake with BNB and directly swap ERC20 tokens without a third party. • When a user unstakes early, they will be charged with 20% of earned ALDN reward tokens as a penalty fee. Rewards will be calculated using the BSC block. • Rewards will be distributed within two years and 30% of total supply will be used for rewards.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**
File 6: NFTStaking.sol • In the NFT staking pool, staked currency will be locked and inaccessible for a specified period. The end of this set time, users can withdraw all ALDN tokens as rewards and transfer them into their own wallets.	**YES, This is valid.**

• 15% of the total supply will go into this staking pool as rewards. Rewards will be distributed within two years.	
File 7: SwapAndLiquify.sol • The owner can add new liquidity.	**YES, This is valid.**

Audit Summary

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



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

We found 0 critical, 0 high, 0 medium and 10 low and some very low level issues.

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

These audit scope have 7 smart contracts. These smart contracts also contain Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

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

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

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

Documentation

We were given MagicLamp protocol smart contracts code in the form of a hash code MD5 and SHA256 format. The hashes of that code are mentioned above in the table.

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

Another source of information was its official website http://magiclamp.gold/ 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 used in external smart contract calls.

AS-IS overview

MagicLamp protocols are smart contracts, having functionality like LP staking, PancakeSwap Liquidity, NFT staking pool, etc.

MagicLamps.sol

(1) Interface

(a) IWETH

(b) IMagicLampWallet

(c) IERC20

(2) Inherited contracts

(a) IMagicLamps

(b) MagicLampsERC721

(3) Struct

(a) Tier

(4) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

(c) using EnumerableSet for EnumerableSet.UintSet;

(d) using EnumerableMap for EnumerableMap.UintToAddressMap;

(e) using Stringstrings for string;

(f) using SafeERC20 for IERC20;

(5) Events

(a) event NameChange (uint256 indexed magicLampIndex, string newName);

(b) event DistributeReferralRewards (uint256 indexed magicLampIndex, address indexed tokenAddress, uint256 amount);

(c) event EarnReferralReward (address indexed account, uint256 amount);

(d) event WithdrawFund (uint256 prizeFund, uint256 liquidityFund, uint256 treasuryFund);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	tokenOfOwnerByIndex	read	Passed	No Issue
2	totalSupply	read	Passed	No Issue
3	tokenByIndex	read	Passed	No Issue
4	tokenNameByIndex	read	Passed	No Issue
5	isNameReserved	read	Passed	No Issue
6	isMintedBeforeReveal	read	Passed	No Issue
7	initMagicLampWalletAddress	write	access by Owner	No Issue
8	distributeReferralRewards	external	access by Owner	No Issue
9	_getTier	internal	Passed	No Issue
10	estimateMagicLampPurchaseAmount	read	Passed	No Issue
11	estimateALDNRewardAmount	read	Passed	No Issue
12	mintMagicLamp	write	External instead of public	Refer Audit Findings
13	withdrawFund	external	Passed	No Issue
14	getRandomNumber	read	Passed	No Issue
15	_rewardReferral	internal	access by Owner	No Issue
16	finalizeStartingIndex	write	Passed	No Issue
17	changeName	write	Passed	No Issue
18	_toggleReserveName	internal	Passed	No Issue
19	validateName	write	Passed	No Issue
20	toLower	write	Passed	No Issue
21	_percent	internal	Passed	No Issue

GenieToken.sol

(1) Inherited contracts

(a) ERC20

(2) Usages

(a) using SafeMath for uint256;

(3) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lastClaim	read	Passed	No Issue
2	accumulated	read	Passed	No Issue

3	claim	write	Infinite loops possibility at multiple places	Refer Audit Findings
4	transferFrom	write	Passed	No Issue
5	setMagicLampAddress	write	Passed	No Issue

MagicLampWallet.sol

(1) Interface

(a) IERC20

(b) IERC721

(c) IERC1155

(d) ISwap

(2) Inherited contracts

(a) Context

(b) Ownable

(c) ERC165

(d) Events

(e) IERC1155Receiver

(f) IERC721Receiver

(3) Struct

(a) Token

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	onlyMagicLampOwner	modifier	Passed	No Issue
2	magicLampExists	modifier	Passed	No Issue
3	unlocked	modifier	Passed	No Issue
4	tokenTypeERC20	external	Passed	No Issue
5	tokenTypeERC721	external	Passed	No Issue
6	tokenTypeERC1155	external	Passed	No Issue
7	existsId	read	Passed	No Issue
8	isLocked	read	Passed	No Issue
9	getInsideTokensCount	read	Passed	No Issue
10	getTokens	read	Passed	No Issue
11	getERC20Tokens	read	Passed	No Issue

12	getERC721Tokens	read	Passed	No Issue
13	getERC721OrERC1155Ids	read	access only Owner	No Issue
14	getERC1155Tokens	read	access only MagicLampOwner	No Issue
15	getERC1155TokenBalances	read	Passed	No Issue
16	setSwap	external	access only Owner	No Issue
17	lockMagicLamp	external	access only MagicLampOwner	No Issue
18	depositErc20IntoMagicLamp	external	Infinite loop possibility	Refer Audit Findings
19	withdrawErc20FromMagicLamp	external	Passed	No Issue
20	sendErc20	write	access only MagicLampOwner	No Issue
21	depositErc721IntoMagicLamp	external	Passed	No Issue
22	withdrawErc721FromMagicLamp	write	access only MagicLampOwner	No Issue
23	sendErc721	write	access only MagicLampOwner	No Issue
24	depositErc1155IntoMagicLamp	external	Passed	No Issue
25	withdrawErc1155FromMagicLamp	write	access only MagicLampOwner	No Issue
26	sendErc1155	write	access only MagicLampOwner	No Issue
27	withdrawAll	external	Passed	No Issue
28	sendAll	external	Passed	No Issue
29	increaseInsideTokenBalance	external	Passed	No Issue
30	swapErc20	external	access only MagicLampOwner	No Issue
31	swapErc721	external	access only MagicLampOwner	No Issue
32	swapErc1155	external	access only MagicLampOwner	No Issue
33	_popERC1155FromMagicLamp	write	Passed	No Issue
34	_increaseInsideTokenBalance	write	Passed	No Issue
35	_increaseInsideERC1155TokenBalance	write	Passed	No Issue
36	_decreaseInsideTokenBalance	write	Passed	No Issue
37	_decreaseInsideERC1155TokenBalance	write	Passed	No Issue
38	_putInsideTokenId	write	Passed	No Issue
39	_putInsideTokenIdForERC1155	write	Passed	No Issue

40	_popInsideTokenId	write	Passed	No Issue
41	_popInsideTokenIdForERC1155	write	Passed	No Issue
42	_putTokenIntoMagicLamp	write	Passed	No Issue
43	_popTokenFromMagicLamp	write	Passed	No Issue
44	onERC1155Received	external	Passed	No Issue
45	onERC1155BatchReceived	external	Passed	No Issue

LPStaking.sol

(1) Interface

(a) IALDN

(2) Inherited contracts

(a) ReentrancyGuard

(b) Ownable

(3) Struct

(a) StakeInfo

(4) Usages

(a) using SafeMath for uint256;

(b) using SafeMath for uint128;

(c) using Address for address;

(5) Events

(a) event Staked(address indexed user, uint256 lpAmount);

(b) event Withdraw(address indexed user, uint256 lpAmount, uint256 tokenAmount);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	setMultipliers	write	access only Owner	No Issue
2	setTokenAddress	write	access only Owner	No Issue
3	setPairAddressAddress	write	access only Owner	No Issue

4	getPeriod	write	Passed	No Issue
5	calcMultiplier	read	Passed	No Issue
6	getALIPerBlockForReward	read	Passed	No Issue
7	getStakedAmount	read	Passed	No Issue
8	getReward	read	Passed	No Issue
9	stake	write	Passed	No Issue
10	withdraw	write	Passed	No Issue

NFTStaking.sol

(1) Interface

(a) IALDN

(2) Inherited contracts

(a) Ownable

(b) ReentrancyGuard

(3) Struct

(a) StakeInfo

(4) Usages

(a) using SafeMath for uint256;

(b) using SafeMath for uint128;

(c) using Address for address;

(5) Events

(a) event Staked(address indexed user, uint256 tokenId);

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

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	setMultipliers	write	access only Owner	No Issue
2	setMagicLampAddress	write	access only Owner	No Issue
3	setTokenAddress	write	access only Owner	No Issue

4	getPeriod	write	Passed	No Issue
5	calcMultiplier	read	Passed	No Issue
6	getALIPerBlockForReward	read	Passed	No Issue
7	getStakedAmount	read	Passed	No Issue
8	getReward	external	Passed	No Issue
9	stake	write	Passed	No Issue
10	withdraw	write	Passed	No Issue

ALDN.sol

(1) Interface

(a) ISwapAndLiquify

(2) Inherited contracts

(a) IERC20

(b) Ownable

(3) Struct

(a) VotesCheckpoint

(b) RateCheckpoint

(4) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

(5) Events

(a) event SwapAndLiquifyAddressChanged(address priviousAddress, address newAddress);

(b) event SwapAndLiquifyEnabledChanged(bool enabled);

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

(d) event DelegateVotesChanged(address indexed delegate, uint previousROwned, uint previousTOwned, uint newROwned, uint newTOwned);

(e) event RateChanged(uint previousRate, uint newRate);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	name	read	Passed	No Issue
2	symbol	read	Passed	No Issue
3	decimals	read	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	balanceOf	read	Passed	No Issue
6	transfer	write	Passed	No Issue
7	allowance	read	Passed	No Issue
8	approve	write	Passed	No Issue
9	transferFrom	write	Passed	No Issue
10	increaseAllowance	write	Passed	No Issue
11	decreaseAllowance	write	Passed	No Issue
12	isExcludedFromReward	read	Passed	No Issue
13	totalFees	read	Passed	No Issue
14	_getOwns	read	Passed	No Issue
15	deliver	write	Passed	No Issue
16	reflectionFromToken	read	Passed	No Issue
17	_tokenFromReflection	write	Passed	No Issue
18	tokenFromReflection	read	Passed	No Issue
19	excludeFromReward	write	access only Owner	No Issue
20	includeInReward	external	Infinite loops possibility at multiple places	Refer Audit Findings
21	_transferBothExcluded	write	Passed	No Issue
22	excludeFromFee	write	access only Owner	No Issue
23	includeInFee	write	access only Owner	No Issue
24	excludeFromMaxTxAmount	write	access only Owner	No Issue
25	includeInMaxTxAmount	write	access only Owner	No Issue
26	setTaxFeePercent	external	access only Owner	No Issue
27	setLiquidityFeePercent	external	access only Owner	No Issue
28	setMaxTxPercent	external	access only Owner	No Issue
29	setSwapAndLiquifyAddress	external	access only Owner	No Issue
30	setSwapAndLiquifyEnabled	external	access only Owner	No Issue
31	_reflectFee	write	Passed	No Issue
32	_getValues	read	Passed	No Issue
33	_getTValues	read	Passed	No Issue

34	_getRValues	write	Passed	No Issue
35	_getCurrentSupply	read	Passed	No Issue
36	_getCurrentRate	read	Passed	No Issue
37	_getPriorRate	read	Passed	No Issue
38	_takeLiquidity	write	Passed	No Issue
39	calculateTaxFee	read	Passed	No Issue
40	calculateLiquidityFee	read	Passed	No Issue
41	removeAllFee	write	Passed	No Issue
42	restoreAllFee	write	Passed	No Issue
43	isExcludedFromFee	read	Passed	No Issue
44	isExcludedFromMaxTxAmount	read	Passed	No Issue
45	_approve	write	Passed	No Issue
46	_transfer	write	Passed	No Issue
47	_tokenTransfer	write	Passed	No Issue
48	_transferStandard	write	Passed	No Issue
49	_transferToExcluded	write	Passed	No Issue
50	_transferFromExcluded	write	Passed	No Issue
51	burn	external	Passed	No Issue
52	delegate	write	Passed	No Issue
53	delegateBySig	write	Passed	No Issue
54	_getVotes	write	Passed	No Issue
55	getCurrentVotes	external	Passed	No Issue
56	getPriorVotes	read	Passed	No Issue
57	_delegate	write	Passed	No Issue
58	_moveDelegates	write	Passed	No Issue
59	_writeCheckpoint	write	Passed	No Issue
60	safe32	write	Passed	No Issue
61	_writeRateCheckpoint	write	Passed	No Issue
62	_getChainId	read	Passed	No Issue

SwapAndLiquify.sol

(1) Interface

(a) IUniswapV2Factory

(b) IUniswapV2Pair

(c) IUniswapV2Router01

(d) IUniswapV2Router02

(2) Inherited contracts

(a) Ownable

(3) Usages

(a) using SafeMath for uint256;

(4) Events

(a) event SwapAndLiquified(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);

(b) event LiquidityInitialized(uint256 tokensIntoLiquidity, uint256 ethIntoLiquidity);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	swapAndLiquify	write	Passed	No Issue
3	swapTokensForEth	write	Passed	No Issue
4	initializeLiquidity	write	Passed	No Issue
5	addLiquidity	write	Passed	No Issue

Severity Definitions

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

Audit Findings

ALDN.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loops possibility at multiple places:

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "ALDN::includeInReward: account is already included");

    (uint256 oldROwned, uint256 oldTOwned) = _getOwns(account);

    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }

    (uint256 newROwned, uint256 newTOwned) = _getOwns(account);

    _moveDelegates(delegates[account], delegates[account], oldROwned, oldTOwned, newROwned, newTOwned);
}
```

There are includeInReward functions in the smart contracts, where array.length is used directly in the loops. 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.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

(2) Make variables constant:

```
string private _name = "Magiclamp Governance Token";
string private _symbol = "ALDN";
uint8 private _decimals = 9;
```

Name, Symbol, Decimals. These variables 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.

SwapAndLiquify.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

GenieToken.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loops possibility at multiple places:

```
function claim(uint256[] memory tokenIndices) public returns (uint256) {
    require(block.timestamp > emissionStart, "Emission has not started yet");

    uint256 totalClaimQty = 0;
    for (uint i = 0; i < tokenIndices.length; i++) {
        // Sanity check for non-minted index
        require(tokenIndices[i] < IMagicLamps(magicLampAddress).totalSupply(), "NFT at index has not been minted yet");
        // Duplicate token index check
        for (uint j = i + 1; j < tokenIndices.length; j++) {
            require(tokenIndices[i] != tokenIndices[j], "Duplicate token index");
        }

        uint tokenIndex = tokenIndices[i];
        require(IMagicLamps(magicLampAddress).ownerOf(tokenIndex) == msg.sender, "Sender is not the owner");

        uint256 claimQty = accumulated(tokenIndex);
        if (claimQty != 0) {
            totalClaimQty = totalClaimQty.add(claimQty);
            _lastClaim[tokenIndex] = block.timestamp;
        }
    }
```

There are claim functions in the smart contracts, where array.length is used directly in the loops. 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.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

(2) Function input parameters lack of check:

```
function claim(uint256[] memory tokenIndices) public returns (uint256) {
    require(block.timestamp > emissionStart, "Emission has not started yet");

    uint256 totalClaimQty = 0;
    for (uint i = 0; i < tokenIndices.length; i++) {
        // Sanity check for non-minted index
        require(tokenIndices[i] < IMagicLamps(magicLampAddress).totalSupply(), "NFT at index has not been minted yet");
        // Duplicate token index check
        for (uint j = i + 1; j < tokenIndices.length; j++) {
            require(tokenIndices[i] != tokenIndices[j], "Duplicate token index");
        }

        uint tokenIndex = tokenIndices[i];
        require(IMagicLamps(magicLampAddress).ownerOf(tokenIndex) == msg.sender, "Sender is not the owner");

        uint256 claimQty = accumulated(tokenIndex);
        if (claimQty != 0) {
            totalClaimQty = totalClaimQty.add(claimQty);
            _lastClaim[tokenIndex] = block.timestamp;
        }
    }
```

There are claim functions, pass single parameter tokenIndices and its use directly in for loop with length, not checked is array or its blank.

Resolution: Check that variable is an array and also it's not empty.

Status: Acknowledged.

MagicLamps.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

(2) HardCoded address:

```
address payable public constant liquidityFundAddress = payable(
    0xf797369bF5A4E539a8E9D7A1Ac1c37F29C6F20D3
);
address payable public constant prizeFundAddress = payable(
    0xf797369bF5A4E539a8E9D7A1Ac1c37F29C6F20D3
);
address payable public constant treasuryFundAddress = payable(
    0x1de179f9B37c7F5E4F54b263189f7bbBF7ae9839
);
```

Smart contract has some hard coded addresses.

Resolution: Owner has to double confirm.

Status: Acknowledged.

(3) External instead of public:

If any function is not called from inside the smart contract, then it is better to declare it as external instead of public. As it saves some gas as well.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Status: Acknowledged.

MagicLampWallet.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loop possibility:

```
function depositErc20IntoMagicLamp(uint256 magicLampId, address[] memory tokens, uint256[] memory amounts)
external magicLampExists(magicLampId) {
    require(tokens.length > 0 && tokens.length == amounts.length);

    for (uint256 i; i < tokens.length; i++) {
        require(tokens[i] != address(0));
        IERC20 iToken = IERC20(tokens[i]);

        uint256 prevBalance = iToken.balanceOf(address(this));
        iToken.transferFrom(msg.sender, address(this), amounts[i]);

        uint256 receivedAmount = iToken.balanceOf(address(this)) - prevBalance;

        _increaseInsideTokenBalance(magicLampId, TOKEN_TYPE_ERC20, tokens[i], receivedAmount);

        emit DepositedErc20IntoMagicLamp(magicLampId, msg.sender, tokens[i], receivedAmount);

    }
}
```

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

Resolution: We suggest checking length.

Status: Acknowledged.

(2) Uses of safeTransferFrom when send token from contract/owner to any user:

```
iToken.safeTransferFrom(address(this), to, tokenId, amount, bytes(""));
```

By using this function to send tokens from the contract itself/owner of the contrat, it will check for the approval of that contract for that token.

Resolution: Use transfer or safeTransfer instead of safeTransferFrom because there is no need for approval as the token owner (contract itself) is sending tokens to users.

Status: Acknowledged.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

LPStaking.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Withdraw/Unstake:

As mentioned in doc, When a user unstakes early, they will be charged with 20% of earned ALDN reward tokens as a penalty fee. But in code we could not find that logic in the withdraw function.

Status: Acknowledged.

(2) Make variables constant:

```
uint256 private _stakedPeriod = 2 * 365 days;
```

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

Resolution: Declare that variable as constant. Just put a constant keyword.

Status: Acknowledged.

(3) Rewards Distribution:

Resolution: As mentioned in doc , 30% of the total supply will go into this staking pool as rewards within 2 years. But we could not find that logic in the code.

Status: Acknowledged.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

NFTStaking.sol

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Withdraw/Unstake:

As mentioned in doc, staked currency will be locked and inaccessible for a specified period. But in the withdraw function code, there is no condition to check the time period.

Status: Acknowledged.

(2) Make variables constant:

```
uint256 private _stakedPeriod = 2 * 365 days;
```

These variables will be unchanged. So, please make it constant. It will save some gas.
Resolution: Declare that variable as constant. Just put a constant keyword.
Status: Acknowledged.

(3) Rewards Distribution:

Resolution: As mentioned in doc , 15% of the total supply will go into this staking pool as rewards within 2 years. But Iwe could not find that logic in the code.
Status: Acknowledged.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

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

Resolution: Please use 0.8.6 which is the latest version.

Status: Acknowledged.

Centralization

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

- initMagicLampWalletAddress: MagicLamp owner can check and update address.
- distributeReferralRewards: The MagicLamp owner can check started and ended sale time and distribute referral rewards amounts.
- setSwap: The MagicLampWallet owner can set swap.
- setMultipliers: The LPStaking owner can set a starting timeblock.
- setTokenAddress: The LPStaking owner can set a token address.
- setPairAddressAddress: The LPStaking owner can set a pair address.
- setMultipliers: The NFTStaking owner can set a starting timeblock.
- setMagicLampAddress: The NFTStaking owner can set a MagicLamp address.
- setTokenAddress: The NFTStaking owner can set a Token address.
- withdrawErc721FromMagicLamp: Only MagicLamp owners can withdraw ERC721.
- sendErc1155L: The MagicLampWallet owner can send ERC1155 tokens from one magicLamp to another magicLamp.
- sendErc721: The MagicLampWallet owner can send ERC721 tokens from one magicLamp to another magicLamp.
- swapErc20: The MagicLampWallet owner can swap ERC20.
- swapErc721: The MagicLampWallet owner can swap ERC721.
- swapErc1155: The MagicLampWallet owner can swap ERC1155.

- excludeFromReward: The ALDN owner can check if the account is already excluded or not.
- includeInReward: The ALDN owner can check if the account is already included or not.
- excludeFromFee: The ALDN owner can check excluded Fee.
- includeInFee: The ALDN owner can check the included fee.
- excludeFromMaxTxAmount: The ALDN owner can check excluded maximum tax accounts.
- includeInMaxTxAmount: The ALDN owner can check the included maximum tax account.
- setTaxFeePercent: The ALDN owner can set tax fee Percentage.
- setLiquidityFeePercent: The ALDN owner can set Liquidity Fee Percentage.
- setMaxTxPercent: The ALDN owner can set maximum Tax Percentage.
- setSwapAndLiquifyAddress: The ALDN owner can set Swap and Liquify Address.
- setSwapAndLiquifyEnabled: The ALDN owner can set swap and Liquify enable.

Conclusion

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

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

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

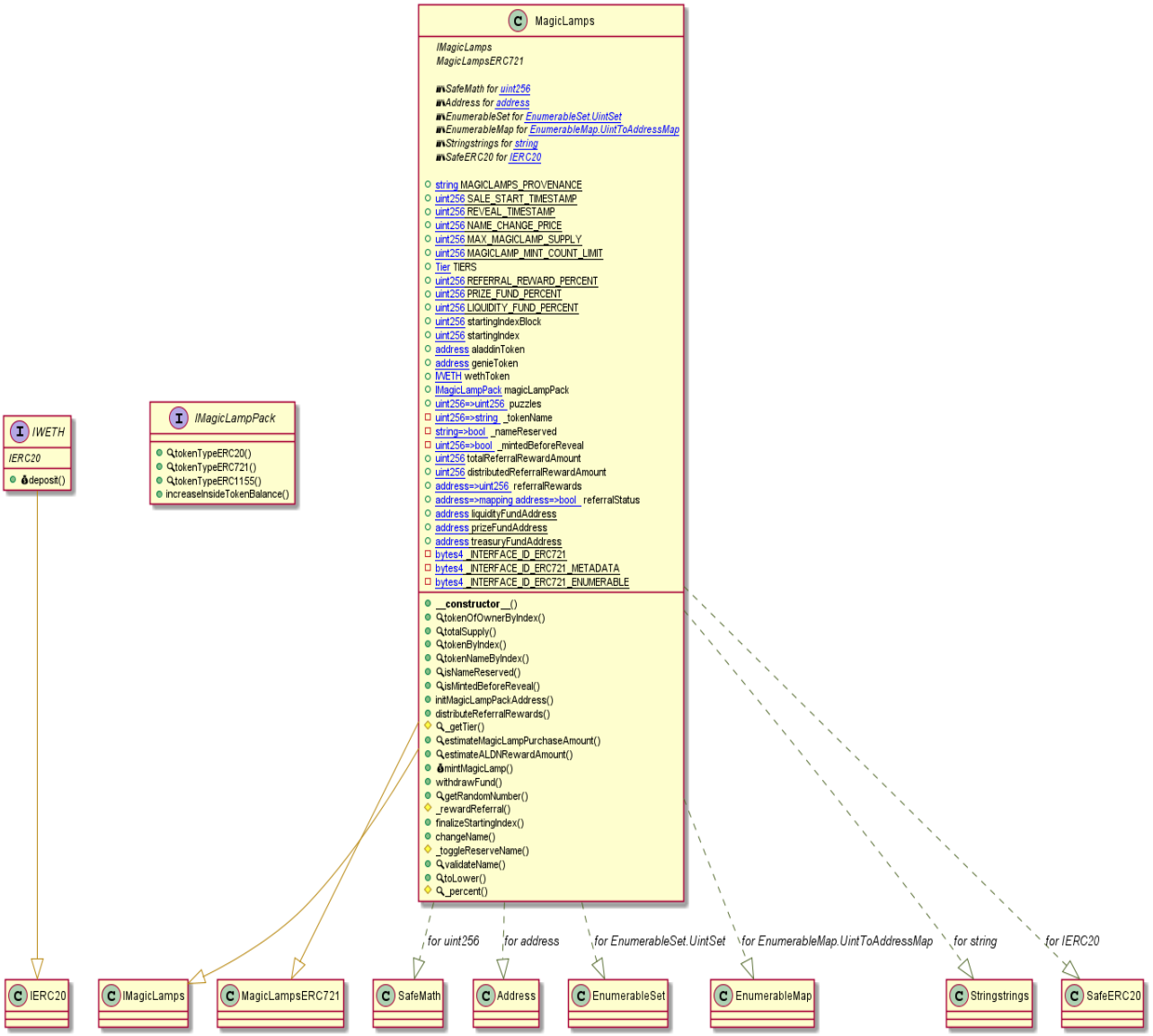
Technical Disclaimer

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

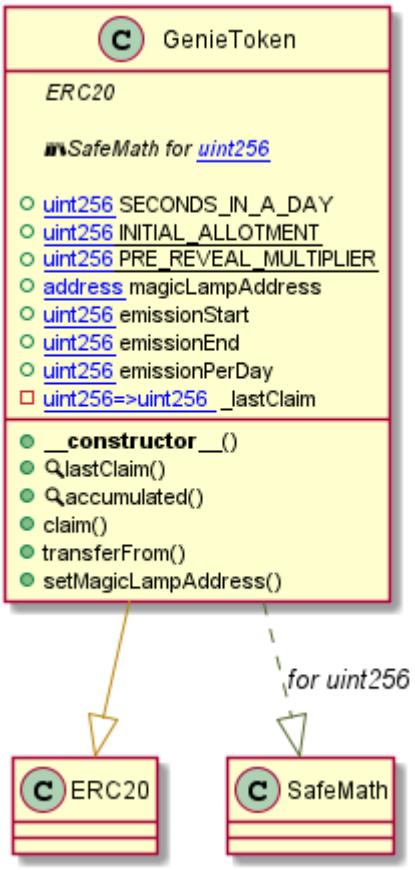
Appendix

Code Flow Diagram - MagicLamp Protocol

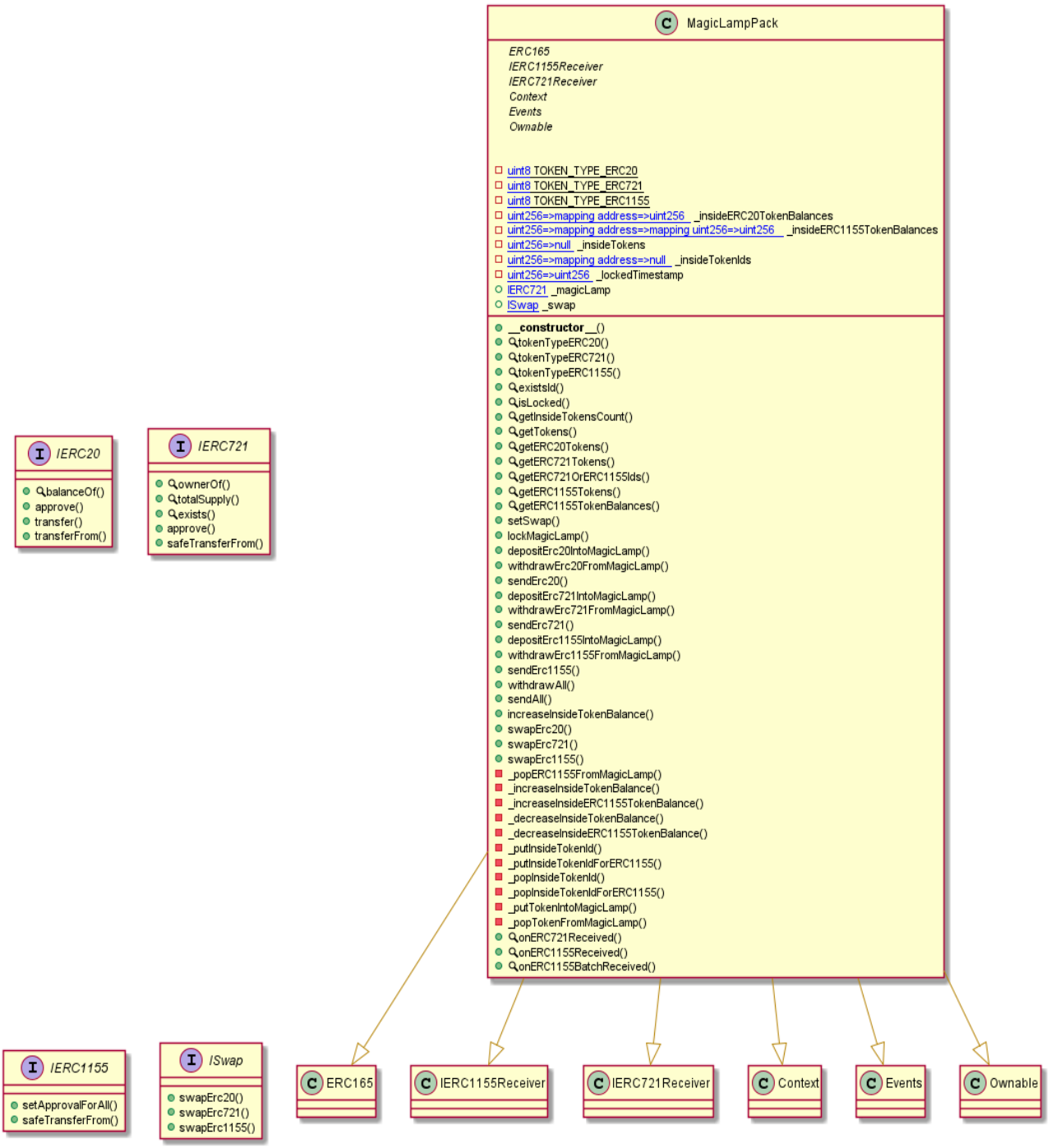
MagicLamps Diagram



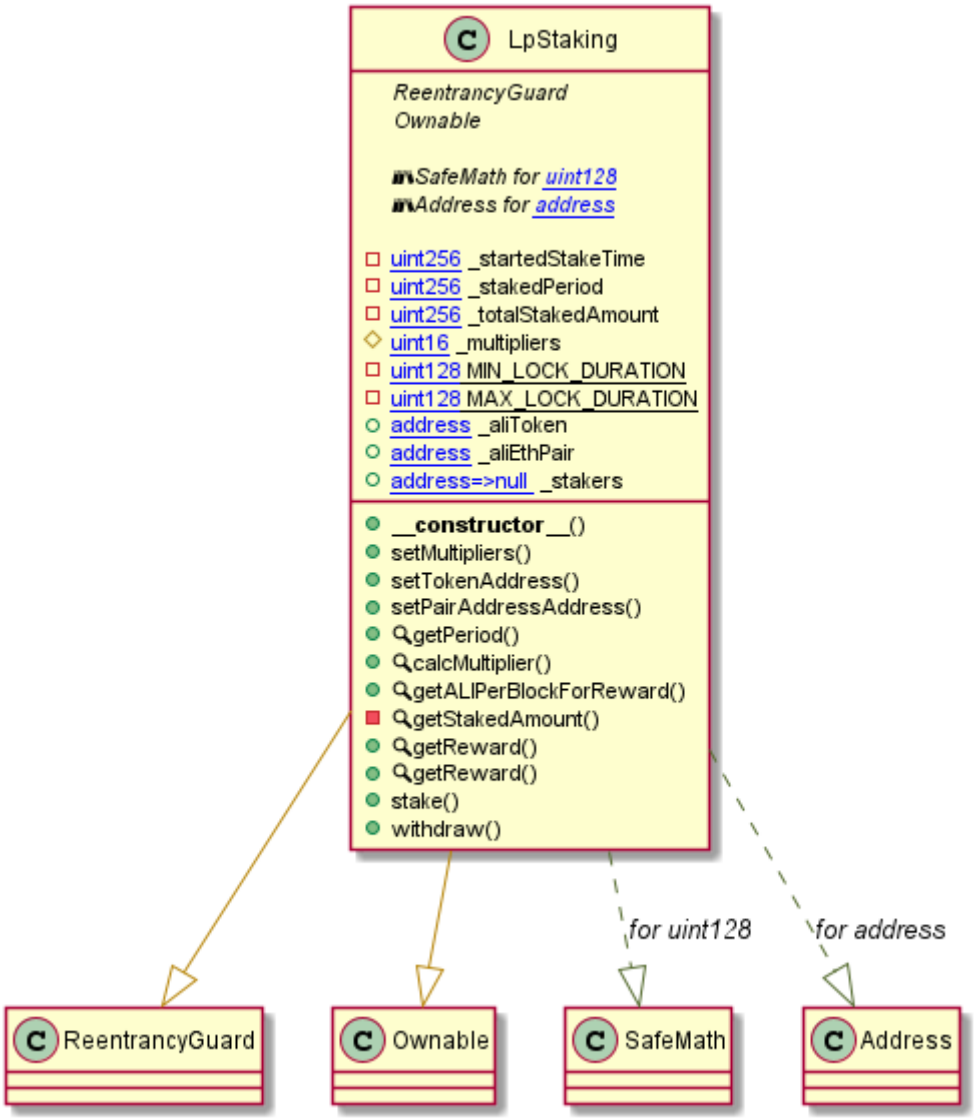
GenieToken Diagram



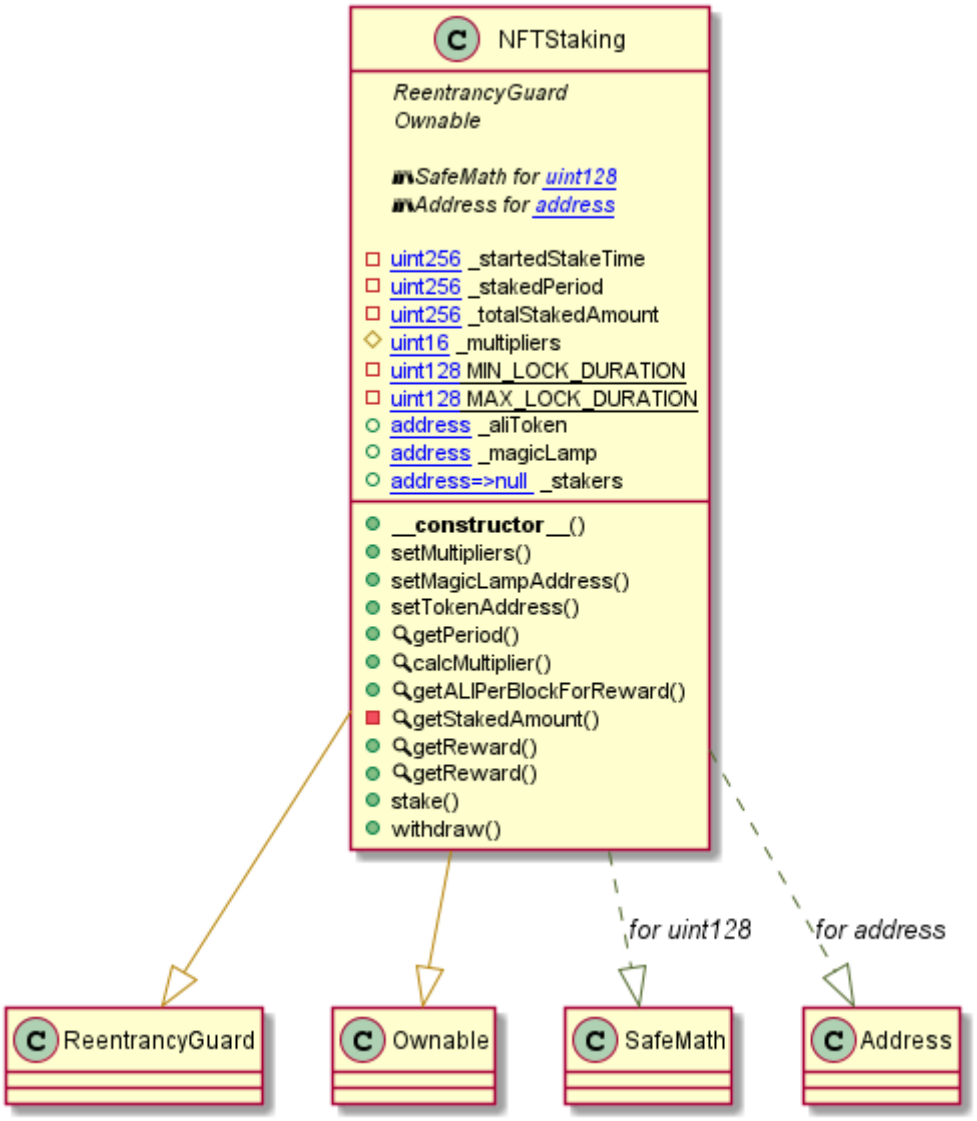
MagicLampWallet Diagram



LPStaking Diagram



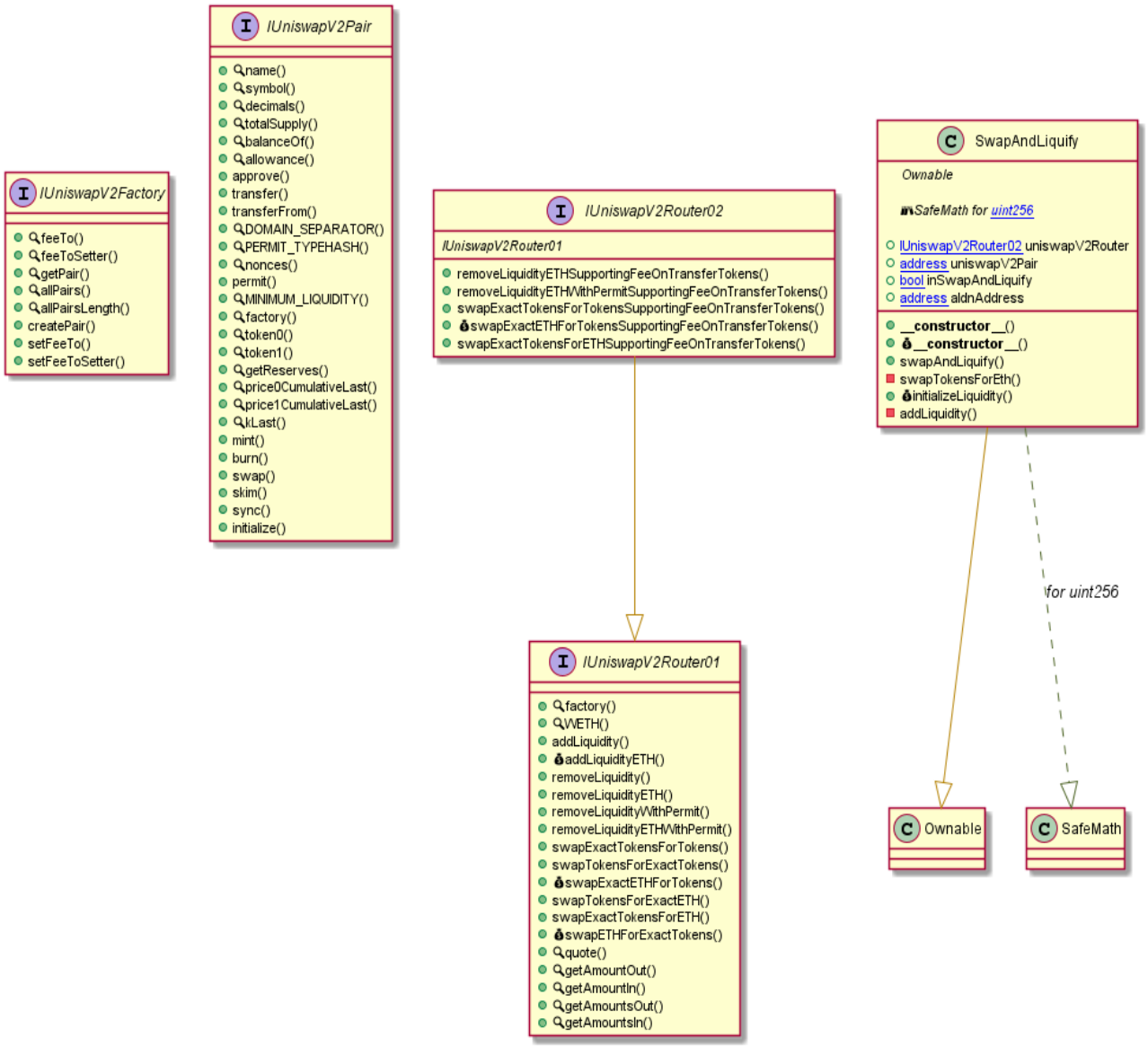
NFTStaking Diagram



ALDN Diagram

Email: audit@EtherAuthority.io

SwapAndLiquify Diagram



Slither Results Log

Slither log >> MagicLamp.sol

```
INFO:Detectors:
MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056) sends eth to arbitrary
user
        Dangerous calls:
        - wethToken.deposit{value: distributionAmount}() (MagicLamps.sol#2052)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinatio
ns
INFO:Detectors:
MagicLamps.getRandomNumber(uint256,uint256) (MagicLamps.sol#2163-2167) uses a weak PRNG:
"(uint256(uint256(keccak256(bytes)(abi.encodePacked(blockhash(uint256)(block.number)))) %
max)).add(min) (MagicLamps.sol#2166)"
MagicLamps.finalizeStartingIndex() (MagicLamps.sol#2188-2201) uses a weak PRNG: "startingIndex =
uint256(blockhash(uint256)(startingIndexBlock)) % MAX_MAGICLAMP_SUPPLY (MagicLamps.sol#2192)"
MagicLamps.finalizeStartingIndex() (MagicLamps.sol#2188-2201) uses a weak PRNG: "startingIndex =
uint256(blockhash(uint256)(block.number - 1)) % MAX_MAGICLAMP_SUPPLY (MagicLamps.sol#2195)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056) ignores return value
by wethToken.transfer(address(magicLampWallet),distributionAmount) (MagicLamps.sol#2053)
MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147) ignores return value by
IALDN(aladdinToken).transfer(_msgSender(),aldnRewardAmount) (MagicLamps.sol#2126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Stringstrings.uint2str(uint256) (MagicLamps.sol#340-360) performs a multiplication on the result of a division:
        -temp = (48 + uint8(_i - _i / 10 * 10)) (MagicLamps.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
MagicLamps.finalizeStartingIndex() (MagicLamps.sol#2188-2201) uses a dangerous strict equality:
        - require(bool,string)(startingIndex == 0,Starting index is already set) (MagicLamps.sol#2189)
MagicLamps.finalizeStartingIndex() (MagicLamps.sol#2188-2201) uses a dangerous strict equality:
        - startingIndex == 0 (MagicLamps.sol#2198)
MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147) uses a dangerous strict
equality:
        - startingIndexBlock == 0 && (totalSupply() == MAX_MAGICLAMP_SUPPLY || block.timestamp >=
REVEAL_TIMESTAMP) (MagicLamps.sol#2140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221):
        External calls:
        - IERC20(genieToken).safeTransferFrom(msg.sender,address(this),NAME_CHANGE_PRICE)
(MagicLamps.sol#2212)
        State variables written after the call(s):
        - _toggleReserveName(_tokenName[tokenId],false) (MagicLamps.sol#2215)
                - _nameReserved[toLower(str)] = isReserve (MagicLamps.sol#2227)
        - _toggleReserveName(newName,true) (MagicLamps.sol#2217)
                - _nameReserved[toLower(str)] = isReserve (MagicLamps.sol#2227)
        - _tokenName[tokenId] = newName (MagicLamps.sol#2218)
Reentrancy in MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056):
        External calls:
        -
magicLampWallet.increaseInsideTokenBalance(i,magicLampWallet.tokenTypeERC20(),address(wethToken),
amount) (MagicLamps.sol#2044)
        State variables written after the call(s):
        - delete referralRewards[owner] (MagicLamps.sol#2046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
```

MagicLamps.validateName(string).i (MagicLamps.sol#2242) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
MagicLampsERC721._mint(address,uint256) (MagicLamps.sol#1700-1710) ignores return value by _holderTokens[to].add(tokenId) (MagicLamps.sol#1706)
MagicLampsERC721._mint(address,uint256) (MagicLamps.sol#1700-1710) ignores return value by _tokenOwners.set(tokenId,to) (MagicLamps.sol#1707)
MagicLampsERC721._burn(uint256) (MagicLamps.sol#1722-1734) ignores return value by _holderTokens[owner].remove(tokenId) (MagicLamps.sol#1730)
MagicLampsERC721._burn(uint256) (MagicLamps.sol#1722-1734) ignores return value by _tokenOwners.remove(tokenId) (MagicLamps.sol#1731)
MagicLampsERC721._transfer(address,address,uint256) (MagicLamps.sol#1747-1761) ignores return value by _holderTokens[from].remove(tokenId) (MagicLamps.sol#1756)
MagicLampsERC721._transfer(address,address,uint256) (MagicLamps.sol#1747-1761) ignores return value by _holderTokens[to].add(tokenId) (MagicLamps.sol#1757)
MagicLampsERC721._transfer(address,address,uint256) (MagicLamps.sol#1747-1761) ignores return value by _tokenOwners.set(tokenId,to) (MagicLamps.sol#1758)
MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes) (MagicLamps.sol#1783-1802) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (MagicLamps.sol#1787-1798)
MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221) ignores return value by IERC20(genieToken).burn(NAME_CHANGE_PRICE) (MagicLamps.sol#2219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
MagicLampsERC721.balanceOf(address).owner (MagicLamps.sol#1501) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLampsERC721.approve(address,uint256).owner (MagicLamps.sol#1556) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLampsERC721.isApprovedForAll(address,address).owner (MagicLamps.sol#1588) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLampsERC721._isApprovedOrOwner(address,uint256).owner (MagicLamps.sol#1661) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLampsERC721._burn(uint256).owner (MagicLamps.sol#1723) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLamps.tokenOfOwnerByIndex(address,uint256).owner (MagicLamps.sol#1983) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
MagicLamps.distributeReferralRewards(uint256,uint256).owner (MagicLamps.sol#2041) shadows:
- Ownable.owner() (MagicLamps.sol#621-623) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (MagicLamps.sol#648) lacks a zero-check on :
- _authorizedNewOwner = authorizedAddress (MagicLamps.sol#649)
MagicLamps.constructor(string,string,address,address,address).aladdin (MagicLamps.sol#1959) lacks a zero-check on :
- aladdinToken = aladdin (MagicLamps.sol#1960)
MagicLamps.constructor(string,string,address,address,address).genie (MagicLamps.sol#1959) lacks a zero-check on :
- genieToken = genie (MagicLamps.sol#1961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056) has external calls inside a loop:
magicLampWallet.increaseInsideTokenBalance(i,magicLampWallet.tokenTypeERC20(),address(wethToken),amount) (MagicLamps.sol#2044)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Variable 'MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes).retval (MagicLamps.sol#1787)' in MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes) (MagicLamps.sol#1783-1802) potentially used before declaration: retval == IERC721Receiver(to).onERC721Received.selector (MagicLamps.sol#1788)
Variable 'MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes).reason (MagicLamps.sol#1789)' in MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes)

(MagicLamps.sol#1783-1802) potentially used before declaration: reason.length == 0
(MagicLamps.sol#1790)
Variable 'MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes).reason
(MagicLamps.sol#1789)' in
MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes)
(MagicLamps.sol#1783-1802) potentially used before declaration: revert(uint256,uint256)(32 +
reason,mload(uint256)(reason)) (MagicLamps.sol#1795)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056):
External calls:
- wethToken.deposit{value: distributionAmount}() (MagicLamps.sol#2052)
- wethToken.transfer(address(magicLampWallet),distributionAmount) (MagicLamps.sol#2053)
External calls sending eth:
- wethToken.deposit{value: distributionAmount}() (MagicLamps.sol#2052)
State variables written after the call(s):
- distributedReferralRewardAmount = distributedReferralRewardAmount.add(distributionAmount)
(MagicLamps.sol#2054)
Reentrancy in MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147):
External calls:
- IALDN(aladdinToken).transfer(_msgSender(),aldnRewardAmount) (MagicLamps.sol#2126)
State variables written after the call(s):
- _mintedBeforeReveal[mintIndex] = true (MagicLamps.sol#2131)
- puzzles[mintIndex] = getRandomNumber(type()(uint256).min,type()(uint256).max.sub(1))
(MagicLamps.sol#2133)
- _rewardReferral(referrer,_msgSender(),msg.value) (MagicLamps.sol#2145)
- referralRewards[referrer] = referralRewards[referrer].add(rewardAmount) (MagicLamps.sol#2175)
- referralRewards[referee] = referralRewards[referee].add(rewardAmount) (MagicLamps.sol#2178)
- _rewardReferral(referrer,_msgSender(),msg.value) (MagicLamps.sol#2145)
- referralStatus[referrer][referee] = true (MagicLamps.sol#2181)
- startingIndexBlock = block.number (MagicLamps.sol#2141)
- _rewardReferral(referrer,_msgSender(),msg.value) (MagicLamps.sol#2145)
- totalReferralRewardAmount = totalReferralRewardAmount.add(rewardAmount)
(MagicLamps.sol#2176)
- totalReferralRewardAmount = totalReferralRewardAmount.add(rewardAmount)
(MagicLamps.sol#2179)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221):
External calls:
- IERC20(genieToken).safeTransferFrom(msg.sender,address(this),NAME_CHANGE_PRICE)
(MagicLamps.sol#2212)
- IERC20(genieToken).burn(NAME_CHANGE_PRICE) (MagicLamps.sol#2219)
Event emitted after the call(s):
- NameChange(tokenId,newName) (MagicLamps.sol#2220)
Reentrancy in MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056):
External calls:
-
magicLampWallet.increaseInsideTokenBalance(i,magicLampWallet.tokenTypeERC20(),address(wethToken),
amount) (MagicLamps.sol#2044)
Event emitted after the call(s):
- DistributeReferralRewards(i,address(wethToken),amount) (MagicLamps.sol#2047)
Reentrancy in MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147):
External calls:
- IALDN(aladdinToken).transfer(_msgSender(),aldnRewardAmount) (MagicLamps.sol#2126)
- _safeMint(_msgSender(),mintIndex) (MagicLamps.sol#2134)
- IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data)
(MagicLamps.sol#1787-1798)
Event emitted after the call(s):
- Transfer(address(0),to,tokenId) (MagicLamps.sol#1709)
- _safeMint(_msgSender(),mintIndex) (MagicLamps.sol#2134)
Reentrancy in MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147):
External calls:
- IALDN(aladdinToken).transfer(_msgSender(),aldnRewardAmount) (MagicLamps.sol#2126)

Event emitted after the call(s):
- EarnReferralReward(referrer,rewardAmount) (MagicLamps.sol#2177)
- _rewardReferral(referrer,_msgSender(),msg.value) (MagicLamps.sol#2145)
- EarnReferralReward(referee,rewardAmount) (MagicLamps.sol#2180)
- _rewardReferral(referrer,_msgSender(),msg.value) (MagicLamps.sol#2145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
MagicLamps.distributeReferralRewards(uint256,uint256) (MagicLamps.sol#2030-2056) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp > SALE_START_TIMESTAMP,Sale has not started) (MagicLamps.sol#2031)
MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp >= SALE_START_TIMESTAMP,Sale has not started) (MagicLamps.sol#2118)
- block.timestamp < REVEAL_TIMESTAMP (MagicLamps.sol#2130)
- startingIndexBlock == 0 && (totalSupply() == MAX_MAGICLAMP_SUPPLY || block.timestamp >= REVEAL_TIMESTAMP) (MagicLamps.sol#2140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MagicLamps.sol#426-435) uses assembly
- INLINE ASM (MagicLamps.sol#433)
Address._verifyCallResult(bool,bytes,string) (MagicLamps.sol#571-588) uses assembly
- INLINE ASM (MagicLamps.sol#580-583)
MagicLampsERC721._checkOnERC721Received(address,address,uint256,bytes) (MagicLamps.sol#1783-1802) uses assembly
- INLINE ASM (MagicLamps.sol#1794-1796)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221) compares to a boolean constant:
-require(bool,string)(sha256(bytes)(bytes(toLower(newName))) == sha256(bytes)(bytes(toLower(_tokenName[tokenId]))) || isNameReserved(newName) == false,Name already reserved) (MagicLamps.sol#2210)
MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221) compares to a boolean constant:
-require(bool,string)(validateName(newName) == true,Not a valid new name) (MagicLamps.sol#2208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (MagicLamps.sol#479-481) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MagicLamps.sol#504-506) is never used and should be removed
Address.functionDelegateCall(address,bytes) (MagicLamps.sol#553-555) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (MagicLamps.sol#563-569) is never used and should be removed
Address.functionStaticCall(address,bytes) (MagicLamps.sol#529-531) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (MagicLamps.sol#539-545) is never used and should be removed
Address.sendValue(address,uint256) (MagicLamps.sol#453-459) is never used and should be removed
Context._msgData() (MagicLamps.sol#596-599) is never used and should be removed
EnumerableMap._get(EnumerableMap.Map,bytes32) (MagicLamps.sol#1144-1148) is never used and should be removed
EnumerableMap._remove(EnumerableMap.Map,bytes32) (MagicLamps.sol#1090-1093) is never used and should be removed
EnumerableMap._tryGet(EnumerableMap.Map,bytes32) (MagicLamps.sol#1128-1135) is never used and should be removed
EnumerableMap.get(EnumerableMap.UintToAddressMap,uint256) (MagicLamps.sol#1234-1236) is never used and should be removed
EnumerableMap.remove(EnumerableMap.UintToAddressMap,uint256) (MagicLamps.sol#1184-1186) is never used and should be removed
EnumerableMap.tryGet(EnumerableMap.UintToAddressMap,uint256) (MagicLamps.sol#1222-1225) is never used and should be removed
EnumerableSet.add(EnumerableSet.AddressSet,address) (MagicLamps.sol#956-958) is never used and should be removed

EnumerableSet.at(EnumerableSet.AddressSet,uint256) (MagicLamps.sol#994-996) is never used and should be removed
EnumerableSet.contains(EnumerableSet.AddressSet,address) (MagicLamps.sol#973-975) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (MagicLamps.sol#1028-1030) is never used and should be removed
EnumerableSet.length(EnumerableSet.AddressSet) (MagicLamps.sol#980-982) is never used and should be removed
EnumerableSet.remove(EnumerableSet.AddressSet,address) (MagicLamps.sol#966-968) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (MagicLamps.sol#912-914) is never used and should be removed
MagicLampsERC721._burn(uint256) (MagicLamps.sol#1722-1734) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (MagicLamps.sol#187-196) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MagicLamps.sol#203-210) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MagicLamps.sol#198-201) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (MagicLamps.sol#172-174) is never used and should be removed
SafeMath.div(uint256,uint256,string) (MagicLamps.sol#1423-1428) is never used and should be removed
SafeMath.mod(uint256,uint256) (MagicLamps.sol#1383-1385) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MagicLamps.sol#1445-1450) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (MagicLamps.sol#1400-1405) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (MagicLamps.sol#1254-1260) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (MagicLamps.sol#1296-1301) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (MagicLamps.sol#1308-1313) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (MagicLamps.sol#1279-1289) is never used and should be removed
SafeMath.trySub(uint256,uint256) (MagicLamps.sol#1267-1272) is never used and should be removed
Strings.toHexString(uint256) (MagicLamps.sol#755-766) is never used and should be removed
Strings.toHexString(uint256,uint256) (MagicLamps.sol#771-781) is never used and should be removed
Stringstrings.fromAddress(address) (MagicLamps.sol#362-373) is never used and should be removed
Stringstrings.strConcat(string,string) (MagicLamps.sol#336-338) is never used and should be removed
Stringstrings.strConcat(string,string,string) (MagicLamps.sol#332-334) is never used and should be removed
Stringstrings.strConcat(string,string,string,string) (MagicLamps.sol#328-330) is never used and should be removed
Stringstrings.strConcat(string,string,string,string,string) (MagicLamps.sol#311-326) is never used and should be removed
Stringstrings.uint2str(uint256) (MagicLamps.sol#340-360) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (MagicLamps.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MagicLamps.sol#453-459):
	- (success) = recipient.call{value: amount}() (MagicLamps.sol#457)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MagicLamps.sol#514-521):
	- (success,returndata) = target.call{value: value}(data) (MagicLamps.sol#519)
Low level call in Address.functionStaticCall(address,bytes,string) (MagicLamps.sol#539-545):
	- (success,returndata) = target.staticcall(data) (MagicLamps.sol#543)
Low level call in Address.functionDelegateCall(address,bytes,string) (MagicLamps.sol#563-569):
	- (success,returndata) = target.delegatecall(data) (MagicLamps.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Stringstrings.strConcat(string,string,string,string,string)._a (MagicLamps.sol#311) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string,string)._b (MagicLamps.sol#311) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string,string)._c (MagicLamps.sol#311) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string,string)._d (MagicLamps.sol#311) is not in mixedCase

Parameter Stringstrings.strConcat(string,string,string,string,string)._e (MagicLamps.sol#311) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string)._a (MagicLamps.sol#328) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string)._b (MagicLamps.sol#328) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string)._c (MagicLamps.sol#328) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string,string)._d (MagicLamps.sol#328) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string)._a (MagicLamps.sol#332) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string)._b (MagicLamps.sol#332) is not in mixedCase
Parameter Stringstrings.strConcat(string,string,string)._c (MagicLamps.sol#332) is not in mixedCase
Parameter Stringstrings.strConcat(string,string)._a (MagicLamps.sol#336) is not in mixedCase
Parameter Stringstrings.strConcat(string,string)._b (MagicLamps.sol#336) is not in mixedCase
Parameter Stringstrings.uint2str(uint256)._i (MagicLamps.sol#340) is not in mixedCase
Constant Strings.alphabet (MagicLamps.sol#725) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter MagicLampsERC721.safeTransferFrom(address,address,uint256,bytes)._data (MagicLamps.sol#1612) is not in mixedCase
Variable MagicLampsERC721._tokenOwners (MagicLamps.sol#1470) is not in mixedCase
Variable MagicLampsERC721._holderTokens (MagicLamps.sol#1473) is not in mixedCase
Variable MagicLampsERC721._tokenApprovals (MagicLamps.sol#1476) is not in mixedCase
Variable MagicLampsERC721._operatorApprovals (MagicLamps.sol#1479) is not in mixedCase
Variable MagicLamps.TIERS (MagicLamps.sol#1873) is not in mixedCase
Constant MagicLamps.liquidityFundAddress (MagicLamps.sol#1905-1907) is not in UPPER_CASE_WITH_UNDERSCORES
Constant MagicLamps.prizeFundAddress (MagicLamps.sol#1908-1910) is not in UPPER_CASE_WITH_UNDERSCORES
Constant MagicLamps.treasuryFundAddress (MagicLamps.sol#1911-1913) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MagicLamps.sol#597)" inContext (MagicLamps.sol#591-600)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in MagicLamps.withdrawFund() (MagicLamps.sol#2152-2161):
	External calls:
	- prizeFundAddress.transfer(prizeFund) (MagicLamps.sol#2155)
	- liquidityFundAddress.transfer(liquidityFund) (MagicLamps.sol#2157)
	- treasuryFundAddress.transfer(treasuryFund) (MagicLamps.sol#2159)
	Event emitted after the call(s):
	- WithdrawFund(prizeFund,liquidityFund,treasuryFund) (MagicLamps.sol#2160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
MagicLamps.constructor(string,string,address,address,address) (MagicLamps.sol#1959-1978) uses literals with too many digits:
	- TIERS.push(Tier(8,11110,11110,100000 * 10 ** 15,uint256(248065092280 * 10 ** 9) / 1)) (MagicLamps.sol#1972)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
authorizedNewOwner() should be declared external:
	- Ownable.authorizedNewOwner() (MagicLamps.sol#636-638)
renounceOwnership(address) should be declared external:
	- Ownable.renounceOwnership(address) (MagicLamps.sol#672-677)
name() should be declared external:
	- MagicLampsERC721.name() (MagicLamps.sol#1516-1518)
symbol() should be declared external:
	- MagicLampsERC721.symbol() (MagicLamps.sol#1523-1525)
tokenURI(uint256) should be declared external:
	- MagicLampsERC721.tokenURI(uint256) (MagicLamps.sol#1530-1537)
setBaseURI(string) should be declared external:
	- MagicLampsERC721.setBaseURI(string) (MagicLamps.sol#1548-1550)
approve(address,uint256) should be declared external:
	- MagicLampsERC721.approve(address,uint256) (MagicLamps.sol#1555-1564)
setApprovalForAll(address,bool) should be declared external:
	- MagicLampsERC721.setApprovalForAll(address,bool) (MagicLamps.sol#1578-1583)
transferFrom(address,address,uint256) should be declared external:
	- MagicLampsERC721.transferFrom(address,address,uint256) (MagicLamps.sol#1595-1600)

safeTransferFrom(address,address,uint256) should be declared external:
- MagicLampsERC721.safeTransferFrom(address,address,uint256) (MagicLamps.sol#1605-1607)
tokenOfOwnerByIndex(address,uint256) should be declared external:
- MagicLamps.tokenOfOwnerByIndex(address,uint256) (MagicLamps.sol#1983-1985)
tokenByIndex(uint256) should be declared external:
- MagicLamps.tokenByIndex(uint256) (MagicLamps.sol#1998-2001)
tokenNameByIndex(uint256) should be declared external:
- MagicLamps.tokenNameByIndex(uint256) (MagicLamps.sol#2006-2008)
isMintedBeforeReveal(uint256) should be declared external:
- MagicLamps.isMintedBeforeReveal(uint256) (MagicLamps.sol#2020-2022)
initMagicLampWalletAddress(address) should be declared external:
- MagicLamps.initMagicLampWalletAddress(address) (MagicLamps.sol#2024-2028)
mintMagicLamp(uint256,address) should be declared external:
- MagicLamps.mintMagicLamp(uint256,address) (MagicLamps.sol#2117-2147)
finalizeStartingIndex() should be declared external:
- MagicLamps.finalizeStartingIndex() (MagicLamps.sol#2188-2201)
changeName(uint256,string) should be declared external:
- MagicLamps.changeName(uint256,string) (MagicLamps.sol#2206-2221)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MagicLamps.sol analyzed (22 contracts with 75 detectors), 144 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Slither log >> GenieToken.sol

INFO:Detectors:
GenieToken.accumulated(uint256) (GenieToken.sol#795-815) uses a dangerous strict equality:
- lastClaimed == emissionStart (GenieToken.sol#809)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
GenieToken.setMagicLampAddress(address)._magicLampAddress (GenieToken.sol#860) lacks a zero-check on :
- magicLampAddress = _magicLampAddress (GenieToken.sol#862)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GenieToken.claim(uint256[]) (GenieToken.sol#820-845) has external calls inside a loop:
require(bool,string)(tokenIndices[i] < IMagicLamps(magicLampAddress).totalSupply(),NFT at index has not been minted yet) (GenieToken.sol#826)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
GenieToken.lastClaim(uint256) (GenieToken.sol#784-790) uses timestamp for comparisons
Dangerous comparisons:
- uint256(_lastClaim[tokenIndex]) != 0 (GenieToken.sol#788)
GenieToken.accumulated(uint256) (GenieToken.sol#795-815) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp > emissionStart,Emission has not started yet) (GenieToken.sol#796)
- lastClaimed >= emissionEnd (GenieToken.sol#803)
- lastClaimed == emissionStart (GenieToken.sol#809)
- block.timestamp < emissionEnd (GenieToken.sol#805)
GenieToken.claim(uint256[]) (GenieToken.sol#820-845) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp > emissionStart,Emission has not started yet) (GenieToken.sol#821)
- claimQty != 0 (GenieToken.sol#836)
- require(bool,string)(totalClaimQty != 0,No accumulated GNI) (GenieToken.sol#842)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
GenieToken.accumulated(uint256) (GenieToken.sol#795-815) compares to a boolean constant:
-IMagicLamps(magicLampAddress).isMintedBeforeReveal(tokenIndex) == true (GenieToken.sol#810)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

```
INFO:Detectors:
Context._msgData() (GenieToken.sol#246-249) is never used and should be removed
SafeMath.div(uint256,uint256,string) (GenieToken.sol#708-713) is never used and should be removed
SafeMath.mod(uint256,uint256) (GenieToken.sol#668-670) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (GenieToken.sol#730-735) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (GenieToken.sol#685-690) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (GenieToken.sol#539-545) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (GenieToken.sol#581-586) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (GenieToken.sol#593-598) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (GenieToken.sol#564-574) is never used and should be removed
SafeMath.trySub(uint256,uint256) (GenieToken.sol#552-557) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (GenieToken.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter GenieToken.setMagicLampAddress(address)._magicLampAddress (GenieToken.sol#860) is not
in mixedCase
Variable GenieToken.SECONDS_IN_A_DAY (GenieToken.sol#748) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GenieToken.sol#247)" inContext (GenieToken.sol#241-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
GenieToken.SECONDS_IN_A_DAY (GenieToken.sol#748) should be constant
GenieToken.emissionPerDay (GenieToken.sol#762) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (GenieToken.sol#280-282)
symbol() should be declared external:
        - ERC20.symbol() (GenieToken.sol#288-290)
decimals() should be declared external:
        - ERC20.decimals() (GenieToken.sol#305-307)
totalSupply() should be declared external:
        - ERC20.totalSupply() (GenieToken.sol#312-314)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (GenieToken.sol#319-321)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (GenieToken.sol#331-334)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (GenieToken.sol#350-353)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (GenieToken.sol#368-378)
        - GenieToken.transferFrom(address,address,uint256) (GenieToken.sol#847-856)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (GenieToken.sol#392-395)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (GenieToken.sol#411-417)
burn(uint256) should be declared external:
        - ERC20.burn(uint256) (GenieToken.sol#419-422)
claim(uint256[]) should be declared external:
        - GenieToken.claim(uint256[]) (GenieToken.sol#820-845)
setMagicLampAddress(address) should be declared external:
        - GenieToken.setMagicLampAddress(address) (GenieToken.sol#860-863)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GenieToken.sol analyzed (9 contracts with 75 detectors), 37 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MagicLampWallet.sol

```
INFO:Detectors:
MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]) (MagicLampWallet.sol#585-602)
ignores return value by iToken.transferFrom(msg.sender,address(this),amounts[i])
(MagicLampWallet.sol#593)
MagicLampWallet.withdrawErc20FromMagicLamp(uint256,address[],uint256[],address)
(MagicLampWallet.sol#607-621) ignores return value by iToken.transfer(to,amounts[i])
(MagicLampWallet.sol#615)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
MagicLampWallet.unlocked(uint256) (MagicLampWallet.sol#403-406) uses a dangerous strict equality:
	- require(bool,string)(_lockedTimestamp[magicLampId] == 0 || _lockedTimestamp[magicLampId] <
block.timestamp,MagicLamp is locked) (MagicLampWallet.sol#404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in
MagicLampWallet.swapErc1155(uint256,address,uint256,uint256,address,uint256,uint8,address)
(MagicLampWallet.sol#861-878):
	External calls:
	- IERC1155(inToken).setApprovalForAll(address(_swap),true) (MagicLampWallet.sol#867)
	- _swap.swapErc1155(magicLampId,inToken,inId,inAmount,outToken,outId,router,to)
(MagicLampWallet.sol#869)
	State variables written after the call(s):
	- _decreaseInsideERC1155TokenBalance(magicLampId,inToken,inId,inAmount)
(MagicLampWallet.sol#873)
		- _insideERC1155TokenBalances[magicLampId][token][tokenId] -= amount
(MagicLampWallet.sol#928)
	- _popInsideTokenIdForERC1155(magicLampId,inToken,inId) (MagicLampWallet.sol#875)
		- delete _insideERC1155TokenBalances[magicLampId][token][tokenId]
(MagicLampWallet.sol#982)
	- _popERC1155FromMagicLamp(magicLampId,inToken,inId) (MagicLampWallet.sol#877)
		- delete _insideERC1155TokenBalances[magicLampId][token][tokenId]
(MagicLampWallet.sol#885)
	- _popInsideTokenIdForERC1155(magicLampId,inToken,inId) (MagicLampWallet.sol#875)
		- ids[i] = ids[ids.length - 1] (MagicLampWallet.sol#965)
		- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#971)
	- _popERC1155FromMagicLamp(magicLampId,inToken,inId) (MagicLampWallet.sol#877)
		- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#886)
Reentrancy in MagicLampWallet.swapErc20(uint256,address,uint256,address,uint8,address)
(MagicLampWallet.sol#833-845):
	External calls:
	- IERC20(inToken).approve(address(_swap),inAmount) (MagicLampWallet.sol#838)
	- _swap.swapErc20(magicLampId,inToken,inAmount,outToken,router,to) (MagicLampWallet.sol#840)
	State variables written after the call(s):
	- _decreaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,inToken,inAmount)
(MagicLampWallet.sol#844)
		- _insideERC20TokenBalances[magicLampId][token] -= amount (MagicLampWallet.sol#914)
		- delete _insideERC20TokenBalances[magicLampId][token] (MagicLampWallet.sol#917)
Reentrancy in MagicLampWallet.swapErc721(uint256,address,uint256,address,uint8,address)
(MagicLampWallet.sol#847-859):
	External calls:
	- IERC721(inToken).approve(address(_swap),inId) (MagicLampWallet.sol#852)
	- _swap.swapErc721(magicLampId,inToken,inId,outToken,router,to) (MagicLampWallet.sol#854)
	State variables written after the call(s):
	- _popInsideTokenId(magicLampId,inToken,inId) (MagicLampWallet.sol#858)
		- ids[i] = ids[ids.length - 1] (MagicLampWallet.sol#965)
		- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#971)
Reentrancy in MagicLampWallet.withdrawAll(uint256,address) (MagicLampWallet.sol#776-797):
	External calls:
	- withdrawErc20FromMagicLamp(magicLampId,erc20Addresses,erc20Balances,to)
(MagicLampWallet.sol#781)
```

- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
- withdrawErc721FromMagicLamp(magicLampId,erc721Addresses[a],ids,to) (MagicLampWallet.sol#787)
- iToken.safeTransferFrom(tokenOwner,to,tokenIds[i]) (MagicLampWallet.sol#675)
State variables written after the call(s):
- withdrawErc721FromMagicLamp(magicLampId,erc721Addresses[a],ids,to) (MagicLampWallet.sol#787)
- _insideERC20TokenBalances[magicLampId][token] -= amount (MagicLampWallet.sol#914)
- delete _insideERC20TokenBalances[magicLampId][token] (MagicLampWallet.sol#917)
- withdrawErc721FromMagicLamp(magicLampId,erc721Addresses[a],ids,to) (MagicLampWallet.sol#787)
- tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
- delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)
Reentrancy in MagicLampWallet.withdrawAll(uint256,address) (MagicLampWallet.sol#776-797):
External calls:
- withdrawErc20FromMagicLamp(magicLampId,erc20Addresses,erc20Balances,to) (MagicLampWallet.sol#781)
- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
- withdrawErc1155FromMagicLamp(magicLampId,erc1155Addresses[a_scope_0],ids_scope_1,tokenBalances,to) (MagicLampWallet.sol#795)
- iToken.safeTransferFrom(address(this),to,tokenId,amount,bytes()) (MagicLampWallet.sol#734)
State variables written after the call(s):
- withdrawErc1155FromMagicLamp(magicLampId,erc1155Addresses[a_scope_0],ids_scope_1,tokenBalances,to) (MagicLampWallet.sol#795)
- tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
- delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
MagicLampWallet.getERC721Tokens(uint256).j (MagicLampWallet.sol#518) is a local variable never initialized
MagicLampWallet.getERC1155Tokens(uint256).j (MagicLampWallet.sol#545) is a local variable never initialized
MagicLampWallet.withdrawErc1155FromMagicLamp(uint256,address,uint256[],uint256[],address).i (MagicLampWallet.sol#730) is a local variable never initialized
MagicLampWallet.getERC20Tokens(uint256).i (MagicLampWallet.sol#499) is a local variable never initialized
MagicLampWallet._putInsideTokenIdForERC1155(uint256,address,uint256).i (MagicLampWallet.sol#946) is a local variable never initialized
MagicLampWallet.depositErc721IntoMagicLamp(uint256,address,uint256[]).i (MagicLampWallet.sol#648) is a local variable never initialized
MagicLampWallet.getInsideTokensCount(uint256).i (MagicLampWallet.sol#459) is a local variable never initialized
MagicLampWallet.withdrawErc721FromMagicLamp(uint256,address,uint256[],address).i (MagicLampWallet.sol#670) is a local variable never initialized
MagicLampWallet.withdrawAll(uint256,address).a (MagicLampWallet.sol#784) is a local variable never initialized
MagicLampWallet.existsId(uint256,address,uint256).i (MagicLampWallet.sol#432) is a local variable never initialized
MagicLampWallet.sendAll(uint256,uint256).a_scope_0 (MagicLampWallet.sol#815) is a local variable never initialized
MagicLampWallet.getERC721Tokens(uint256).i (MagicLampWallet.sol#520) is a local variable never initialized
MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]).i (MagicLampWallet.sol#588) is a local variable never initialized
MagicLampWallet.sendErc20(uint256,address[],uint256[],uint256).i (MagicLampWallet.sol#631) is a local variable never initialized
MagicLampWallet.withdrawErc20FromMagicLamp(uint256,address[],uint256[],address).i (MagicLampWallet.sol#611) is a local variable never initialized
MagicLampWallet._popTokenFromMagicLamp(uint256,uint8,address).i (MagicLampWallet.sol#1018) is a local variable never initialized
MagicLampWallet.sendErc1155(uint256,address,uint256[],uint256[],uint256).i (MagicLampWallet.sol#755) is a local variable never initialized
MagicLampWallet.getERC1155Tokens(uint256).i (MagicLampWallet.sol#547) is a local variable never initialized

MagicLampWallet._popInsideTokenId(uint256,address,uint256).i (MagicLampWallet.sol#963) is a local variable never initialized
MagicLampWallet.depositErc1155IntoMagicLamp(uint256,address,uint256[],uint256[]).i (MagicLampWallet.sol#711) is a local variable never initialized
MagicLampWallet.withdrawAll(uint256,address).a_scope_0 (MagicLampWallet.sol#791) is a local variable never initialized
MagicLampWallet.sendErc721(uint256,address,uint256[],uint256).i (MagicLampWallet.sol#693) is a local variable never initialized
MagicLampWallet.getTokens(uint256).i (MagicLampWallet.sol#482) is a local variable never initialized
MagicLampWallet._putInsideTokenIdForERC1155(uint256,address,uint256).isExist (MagicLampWallet.sol#944) is a local variable never initialized
MagicLampWallet.getERC20Tokens(uint256).j (MagicLampWallet.sol#497) is a local variable never initialized
MagicLampWallet.sendAll(uint256,uint256).a (MagicLampWallet.sol#808) is a local variable never initialized
MagicLampWallet.getERC1155TokenBalances(uint256,address,uint256[]).i (MagicLampWallet.sol#561) is a local variable never initialized
MagicLampWallet._putTokenIntoMagicLamp(uint256,uint8,address).i (MagicLampWallet.sol#994) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
MagicLampWallet.swapErc20(uint256,address,uint256,address,uint8,address) (MagicLampWallet.sol#833-845) ignores return value by IERC20(inToken).approve(address(_swap),inAmount) (MagicLampWallet.sol#838)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (MagicLampWallet.sol#301) lacks a zero-check on :
	- _authorizedNewOwner = authorizedAddress (MagicLampWallet.sol#302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]) (MagicLampWallet.sol#585-602) has external calls inside a loop: prevBalance = iToken.balanceOf(address(this)) (MagicLampWallet.sol#592)
MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]) (MagicLampWallet.sol#585-602) has external calls inside a loop: iToken.transferFrom(msg.sender,address(this),amounts[i]) (MagicLampWallet.sol#593)
MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]) (MagicLampWallet.sol#585-602) has external calls inside a loop: receivedAmount = iToken.balanceOf(address(this)) - prevBalance (MagicLampWallet.sol#595)
MagicLampWallet.withdrawErc20FromMagicLamp(uint256,address[],uint256[],address) (MagicLampWallet.sol#607-621) has external calls inside a loop: iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
MagicLampWallet.sendErc20(uint256,address[],uint256[],uint256) (MagicLampWallet.sol#626-640) has external calls inside a loop: require(bool)(_magicLamp.exists(toMagicLampId)) (MagicLampWallet.sol#633)
MagicLampWallet.depositErc721IntoMagicLamp(uint256,address,uint256[]) (MagicLampWallet.sol#645-660) has external calls inside a loop: iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i]) (MagicLampWallet.sol#653)
MagicLampWallet.withdrawErc721FromMagicLamp(uint256,address,uint256[],address) (MagicLampWallet.sol#665-682) has external calls inside a loop: tokenOwner = iToken.ownerOf(tokenIds[i]) (MagicLampWallet.sol#671)
MagicLampWallet.withdrawErc721FromMagicLamp(uint256,address,uint256[],address) (MagicLampWallet.sol#665-682) has external calls inside a loop: iToken.safeTransferFrom(tokenOwner,to,tokenIds[i]) (MagicLampWallet.sol#675)
MagicLampWallet.depositErc1155IntoMagicLamp(uint256,address,uint256[],uint256[]) (MagicLampWallet.sol#707-720) has external calls inside a loop: iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i],amounts[i],bytes()) (MagicLampWallet.sol#712)
MagicLampWallet.withdrawErc1155FromMagicLamp(uint256,address,uint256[],uint256[],address) (MagicLampWallet.sol#725-744) has external calls inside a loop: iToken.safeTransferFrom(address(this),to,tokenId,amount,bytes()) (MagicLampWallet.sol#734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in MagicLampWallet.depositErc1155IntoMagicLamp(uint256,address,uint256[],uint256[]) (MagicLampWallet.sol#707-720):
	External calls:
	- iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i],amounts[i],bytes()) (MagicLampWallet.sol#712)

State variables written after the call(s):
- _increaseInsideERC1155TokenBalance(magicLampId,TOKEN_TYPE_ERC1155,token,tokenIds[i],amounts[i]) (MagicLampWallet.sol#716)
    - _insideERC1155TokenBalances[magicLampId][token][tokenId] += amount (MagicLampWallet.sol#904)
- _putInsideTokenIdForERC1155(magicLampId,token,tokenIds[i]) (MagicLampWallet.sol#714)
    - ids.push(tokenId) (MagicLampWallet.sol#953)
- _increaseInsideERC1155TokenBalance(magicLampId,TOKEN_TYPE_ERC1155,token,tokenIds[i],amounts[i]) (MagicLampWallet.sol#716)
    - tokens.push(Token(tokenType,tokenAddress)) (MagicLampWallet.sol#1005-1008)

Reentrancy in MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[]) (MagicLampWallet.sol#585-602):

External calls:
- iToken.transferFrom(msg.sender,address(this),amounts[i]) (MagicLampWallet.sol#593)

State variables written after the call(s):
- _increaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,tokens[i],receivedAmount) (MagicLampWallet.sol#597)
    - _insideERC20TokenBalances[magicLampId][token] += amount (MagicLampWallet.sol#896)
- _increaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,tokens[i],receivedAmount) (MagicLampWallet.sol#597)
    - tokens.push(Token(tokenType,tokenAddress)) (MagicLampWallet.sol#1005-1008)

Reentrancy in MagicLampWallet.depositErc721IntoMagicLamp(uint256,address,uint256[]) (MagicLampWallet.sol#645-660):

External calls:
- iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i]) (MagicLampWallet.sol#653)

State variables written after the call(s):
- _putInsideTokenId(magicLampId,token,tokenIds[i]) (MagicLampWallet.sol#655)
    - ids.push(tokenId) (MagicLampWallet.sol#936)

Reentrancy in MagicLampWallet.swapErc1155(uint256,address,uint256,uint256,address,uint256,uint8,address) (MagicLampWallet.sol#861-878):

External calls:
- IERC1155(inToken).setApprovalForAll(address(_swap),true) (MagicLampWallet.sol#867)
- _swap.swapErc1155(magicLampId,inToken,inId,inAmount,outToken,outId,router,to) (MagicLampWallet.sol#869)

State variables written after the call(s):
- _popERC1155FromMagicLamp(magicLampId,inToken,inId) (MagicLampWallet.sol#877)
    - tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
    - delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)

Reentrancy in MagicLampWallet.swapErc20(uint256,address,uint256,address,uint8,address) (MagicLampWallet.sol#833-845):

External calls:
- IERC20(inToken).approve(address(_swap),inAmount) (MagicLampWallet.sol#838)
- _swap.swapErc20(magicLampId,inToken,inAmount,outToken,router,to) (MagicLampWallet.sol#840)

State variables written after the call(s):
- _decreaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,inToken,inAmount) (MagicLampWallet.sol#844)
    - tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
    - delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)

Reentrancy in MagicLampWallet.withdrawErc1155FromMagicLamp(uint256,address,uint256[],uint256[],address) (MagicLampWallet.sol#725-744):

External calls:
- iToken.safeTransferFrom(address(this),to,tokenId,amount,bytes()) (MagicLampWallet.sol#734)

State variables written after the call(s):
- _decreaseInsideERC1155TokenBalance(magicLampId,token,tokenId,amount) (MagicLampWallet.sol#736)
    - _insideERC1155TokenBalances[magicLampId][token][tokenId] -= amount (MagicLampWallet.sol#928)
- _popInsideTokenIdForERC1155(magicLampId,token,tokenId) (MagicLampWallet.sol#738)
    - delete _insideERC1155TokenBalances[magicLampId][token][tokenId] (MagicLampWallet.sol#982)
- _popERC1155FromMagicLamp(magicLampId,token,tokenId) (MagicLampWallet.sol#740)

- delete _insideERC1155TokenBalances[magicLampId][token][tokenId]
(MagicLampWallet.sol#885)
- _popInsideTokenIdForERC1155(magicLampId,token,tokenId) (MagicLampWallet.sol#738)
- ids[i] = ids[ids.length - 1] (MagicLampWallet.sol#965)
- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#971)
- _popERC1155FromMagicLamp(magicLampId,token,tokenId) (MagicLampWallet.sol#740)
- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#886)
- _popERC1155FromMagicLamp(magicLampId,token,tokenId) (MagicLampWallet.sol#740)
- tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
- delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)
Reentrancy in MagicLampWallet.withdrawErc20FromMagicLamp(uint256,address[],uint256[],address)
(MagicLampWallet.sol#607-621):
External calls:
- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
State variables written after the call(s):
- _decreaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,tokens[i],amounts[i])
(MagicLampWallet.sol#617)
- _insideERC20TokenBalances[magicLampId][token] -= amount (MagicLampWallet.sol#914)
- delete _insideERC20TokenBalances[magicLampId][token] (MagicLampWallet.sol#917)
- _decreaseInsideTokenBalance(magicLampId,TOKEN_TYPE_ERC20,tokens[i],amounts[i])
(MagicLampWallet.sol#617)
- tokens[i] = tokens[tokens.length - 1] (MagicLampWallet.sol#1023)
- delete _insideTokens[magicLampId] (MagicLampWallet.sol#1030)
Reentrancy in MagicLampWallet.withdrawErc721FromMagicLamp(uint256,address,uint256[],address)
(MagicLampWallet.sol#665-682):
External calls:
- iToken.safeTransferFrom(tokenOwner,to,tokenIds[i]) (MagicLampWallet.sol#675)
State variables written after the call(s):
- _popInsideTokenId(magicLampId,token,tokenIds[i]) (MagicLampWallet.sol#677)
- ids[i] = ids[ids.length - 1] (MagicLampWallet.sol#965)
- delete _insideTokenIds[magicLampId][token] (MagicLampWallet.sol#971)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MagicLampWallet.depositErc1155IntoMagicLamp(uint256,address,uint256[],uint256[])
(MagicLampWallet.sol#707-720):
External calls:
- iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i],amounts[i],bytes())
(MagicLampWallet.sol#712)
Event emitted after the call(s):
- DepositedErc1155IntoMagicLamp(magicLampId,msg.sender,token,tokenIds[i],amounts[i])
(MagicLampWallet.sol#718)
Reentrancy in MagicLampWallet.depositErc20IntoMagicLamp(uint256,address[],uint256[])
(MagicLampWallet.sol#585-602):
External calls:
- iToken.transferFrom(msg.sender,address(this),amounts[i]) (MagicLampWallet.sol#593)
Event emitted after the call(s):
- DepositedErc20IntoMagicLamp(magicLampId,msg.sender,tokens[i],receivedAmount)
(MagicLampWallet.sol#599)
Reentrancy in MagicLampWallet.depositErc721IntoMagicLamp(uint256,address,uint256[])
(MagicLampWallet.sol#645-660):
External calls:
- iToken.safeTransferFrom(msg.sender,address(this),tokenIds[i]) (MagicLampWallet.sol#653)
Event emitted after the call(s):
- DepositedErc721IntoMagicLamp(magicLampId,msg.sender,token,tokenIds[i])
(MagicLampWallet.sol#657)
Reentrancy in
MagicLampWallet.swapErc1155(uint256,address,uint256,uint256,address,uint256,uint8,address)
(MagicLampWallet.sol#861-878):
External calls:
- IERC1155(inToken).setApprovalForAll(address(_swap),true) (MagicLampWallet.sol#867)
- _swap.swapErc1155(magicLampId,inToken,inId,inAmount,outToken,outId,router,to)
(MagicLampWallet.sol#869)
Event emitted after the call(s):
- SwapedErc1155(msg.sender,magicLampId,inToken,inId,inAmount,outToken,outId,to)
(MagicLampWallet.sol#871)

Reentrancy in MagicLampWallet.swapErc20(uint256,address,uint256,address,uint8,address) (MagicLampWallet.sol#833-845):
External calls:
- IERC20(inToken).approve(address(_swap),inAmount) (MagicLampWallet.sol#838)
- _swap.swapErc20(magicLampId,inToken,inAmount,outToken,router,to) (MagicLampWallet.sol#840)
Event emitted after the call(s):
- SwapedErc20(msg.sender,magicLampId,inToken,inAmount,outToken,to) (MagicLampWallet.sol#842)
Reentrancy in MagicLampWallet.swapErc721(uint256,address,uint256,address,uint8,address) (MagicLampWallet.sol#847-859):
External calls:
- IERC721(inToken).approve(address(_swap),inId) (MagicLampWallet.sol#852)
- _swap.swapErc721(magicLampId,inToken,inId,outToken,router,to) (MagicLampWallet.sol#854)
Event emitted after the call(s):
- SwapedErc721(msg.sender,magicLampId,inToken,inId,outToken,to) (MagicLampWallet.sol#856)
Reentrancy in MagicLampWallet.withdrawAll(uint256,address) (MagicLampWallet.sol#776-797):
External calls:
- withdrawErc20FromMagicLamp(magicLampId,erc20Addresses,erc20Balances,to) (MagicLampWallet.sol#781)
- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
- withdrawErc721FromMagicLamp(magicLampId,erc721Addresses[a],ids,to) (MagicLampWallet.sol#787)
- iToken.safeTransferFrom(tokenOwner,to,tokenIds[i]) (MagicLampWallet.sol#675)
Event emitted after the call(s):
- WithdrewErc721FromMagicLamp(magicLampId,msg.sender,token,tokenIds[i],to) (MagicLampWallet.sol#679)
- withdrawErc721FromMagicLamp(magicLampId,erc721Addresses[a],ids,to) (MagicLampWallet.sol#787)
Reentrancy in MagicLampWallet.withdrawAll(uint256,address) (MagicLampWallet.sol#776-797):
External calls:
- withdrawErc20FromMagicLamp(magicLampId,erc20Addresses,erc20Balances,to) (MagicLampWallet.sol#781)
- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
- withdrawErc1155FromMagicLamp(magicLampId,erc1155Addresses[a_scope_0],ids_scope_1,tokenBalances,to) (MagicLampWallet.sol#795)
- iToken.safeTransferFrom(address(this),to,tokenId,amount,bytes()) (MagicLampWallet.sol#734)
Event emitted after the call(s):
- WithdrewErc1155FromMagicLamp(magicLampId,msg.sender,token,tokenId,amount,to) (MagicLampWallet.sol#742)
- withdrawErc1155FromMagicLamp(magicLampId,erc1155Addresses[a_scope_0],ids_scope_1,tokenBalances,to) (MagicLampWallet.sol#795)
Reentrancy in MagicLampWallet.withdrawErc1155FromMagicLamp(uint256,address,uint256[],uint256[],address) (MagicLampWallet.sol#725-744):
External calls:
- iToken.safeTransferFrom(address(this),to,tokenId,amount,bytes()) (MagicLampWallet.sol#734)
Event emitted after the call(s):
- WithdrewErc1155FromMagicLamp(magicLampId,msg.sender,token,tokenId,amount,to) (MagicLampWallet.sol#742)
Reentrancy in MagicLampWallet.withdrawErc20FromMagicLamp(uint256,address[],uint256[],address) (MagicLampWallet.sol#607-621):
External calls:
- iToken.transfer(to,amounts[i]) (MagicLampWallet.sol#615)
Event emitted after the call(s):
- WithdrewErc20FromMagicLamp(magicLampId,msg.sender,tokens[i],amounts[i],to) (MagicLampWallet.sol#619)
Reentrancy in MagicLampWallet.withdrawErc721FromMagicLamp(uint256,address,uint256[],address) (MagicLampWallet.sol#665-682):
External calls:
- iToken.safeTransferFrom(tokenOwner,to,tokenIds[i]) (MagicLampWallet.sol#675)
Event emitted after the call(s):
- WithdrewErc721FromMagicLamp(magicLampId,msg.sender,token,tokenIds[i],to) (MagicLampWallet.sol#679)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```
INFO:Detectors:
MagicLampWallet.isLocked(uint256) (MagicLampWallet.sol#444-451) uses timestamp for comparisons
        Dangerous comparisons:
        - _lockedTimestamp[magicLampId] == 0 || _lockedTimestamp[magicLampId] < block.timestamp
(MagicLampWallet.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (MagicLampWallet.sol#249-252) is never used and should be removed
ERC165._registerInterface(bytes4) (MagicLampWallet.sol#174-177) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (MagicLampWallet.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
MagicLampWallet (MagicLampWallet.sol#362-1046) should inherit from ISwap
(MagicLampWallet.sol#355-360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Variable MagicLampWallet._magicLamp (MagicLampWallet.sol#389) is not in mixedCase
Variable MagicLampWallet._swap (MagicLampWallet.sol#390) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MagicLampWallet.sol#250)" inContext (MagicLampWallet.sol#244-253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
supportsInterface(bytes4) should be declared external:
        - ERC165.supportsInterface(bytes4) (MagicLampWallet.sol#159-161)
authorizedNewOwner() should be declared external:
        - Ownable.authorizedNewOwner() (MagicLampWallet.sol#289-291)
renounceOwnership(address) should be declared external:
        - Ownable.renounceOwnership(address) (MagicLampWallet.sol#325-330)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MagicLampWallet.sol analyzed (12 contracts with 75 detectors), 79 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ALDN.sol

```
INFO:Detectors:
ALDN.numRateCheckpoints (ALDN.sol#638) is never initialized. It is used in:
        - ALDN._getPriorRate(uint256) (ALDN.sol#929-958)
        - ALDN._moveDelegates(address,address,uint256,uint256,uint256,uint256) (ALDN.sol#1244-1283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
ALDN._writeCheckpoint(address,uint32,uint256,uint256,uint256,uint256) (ALDN.sol#1285-1297) uses a
dangerous strict equality:
        - nCheckpoints > 0 && votesCheckpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber
(ALDN.sol#1288)
ALDN._writeRateCheckpoint(uint32,uint256,uint256) (ALDN.sol#1299-1311) uses a dangerous strict
equality:
        - nCheckpoints > 0 && rateCheckpoints[nCheckpoints - 1].fromBlock == blockNumber (ALDN.sol#1302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in ALDN._transfer(address,address,uint256) (ALDN.sol#1008-1034):
        External calls:
        - overMinTokenBalance && from != owner() && from != address(swapAndLiquify) && !
swapAndLiquify.inSwapAndLiquify() && swapAndLiquifyEnabled (ALDN.sol#1021)
        - swapAndLiquify.swapAndLiquify(contractTokenBalance) (ALDN.sol#1024)
        State variables written after the call(s):
```

- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (ALDN.sol#963)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (ALDN.sol#1082)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (ALDN.sol#1099)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (ALDN.sol#1118)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (ALDN.sol#819)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (ALDN.sol#1083)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (ALDN.sol#1101)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (ALDN.sol#1119)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (ALDN.sol#821)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _rTotal = _rTotal.sub(rFee) (ALDN.sol#873)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (ALDN.sol#965)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (ALDN.sol#1117)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (ALDN.sol#818)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (ALDN.sol#1100)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (ALDN.sol#820)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ALDN.allowance(address,address).owner (ALDN.sol#693) shadows:
- Ownable.owner() (ALDN.sol#424-426) (function)
ALDN._approve(address,address,uint256).owner (ALDN.sol#999) shadows:
- Ownable.owner() (ALDN.sol#424-426) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (ALDN.sol#451) lacks a zero-check on :
- _authorizedNewOwner = authorizedAddress (ALDN.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ALDN._transfer(address,address,uint256) (ALDN.sol#1008-1034):
External calls:
- overMinTokenBalance && from != owner() && from != address(swapAndLiquify) && !swapAndLiquify.inSwapAndLiquify() && swapAndLiquifyEnabled (ALDN.sol#1021)
- swapAndLiquify.swapAndLiquify(contractTokenBalance) (ALDN.sol#1024)
State variables written after the call(s):
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _liquidityFee = _previousLiquidityFee (ALDN.sol#988)
    - _liquidityFee = 0 (ALDN.sol#983)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _previousLiquidityFee = _liquidityFee (ALDN.sol#980)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _previousTaxFee = _taxFee (ALDN.sol#979)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _tFeeTotal = _tFeeTotal.add(tFee) (ALDN.sol#874)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - _taxFee = _previousTaxFee (ALDN.sol#987)
    - _taxFee = 0 (ALDN.sol#982)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - numVotesCheckpoints[delegatee] = nCheckpoints + 1 (ALDN.sol#1293)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - rateCheckpoints[nCheckpoints - 1].rate = newRate (ALDN.sol#1303)
    - rateCheckpoints[nCheckpoints].fromBlock = blockNumber (ALDN.sol#1305)
    - rateCheckpoints[nCheckpoints].rate = newRate (ALDN.sol#1306)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
    - votesCheckpoints[delegatee][nCheckpoints - 1].tOwned = uint96(newTOwned) (ALDN.sol#1289)
    - votesCheckpoints[delegatee][nCheckpoints - 1].rOwned = newROwned (ALDN.sol#1290)
    - votesCheckpoints[delegatee][nCheckpoints] = VotesCheckpoint(blockNumber,uint96(newTOwned),newROwned) (ALDN.sol#1292)

Reentrancy in ALDN.transferFrom(address,address,uint256) (ALDN.sol#702-709):
External calls:
- _transfer(sender,recipient,amount) (ALDN.sol#703)
    - overMinTokenBalance && from != owner() && from != address(swapAndLiquify) && !swapAndLiquify.inSwapAndLiquify() && swapAndLiquifyEnabled (ALDN.sol#1021)
    - swapAndLiquify.swapAndLiquify(contractTokenBalance) (ALDN.sol#1024)

State variables written after the call(s):
- _approve(sender,_msgSender(),spenderAllowance.sub(amount,ERC20: transfer amount exceeds allowance)) (ALDN.sol#706)
- _allowances[owner][spender] = amount (ALDN.sol#1003)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ALDN._transfer(address,address,uint256) (ALDN.sol#1008-1034):
External calls:
- overMinTokenBalance && from != owner() && from != address(swapAndLiquify) && ! swapAndLiquify.inSwapAndLiquify() && swapAndLiquifyEnabled (ALDN.sol#1021)
- swapAndLiquify.swapAndLiquify(contractTokenBalance) (ALDN.sol#1024)

Event emitted after the call(s):
- DelegateVotesChanged(delegatee,oldROwned,oldTOwned,newROwned,newTOwned) (ALDN.sol#1296)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- RateChanged(oldRate,newRate) (ALDN.sol#1310)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- Transfer(sender,recipient,amount) (ALDN.sol#1038)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- Transfer(sender,recipient,tTransferAmount) (ALDN.sol#1087)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- Transfer(sender,recipient,tTransferAmount) (ALDN.sol#1122)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- Transfer(sender,recipient,tTransferAmount) (ALDN.sol#1105)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)
- Transfer(sender,recipient,tTransferAmount) (ALDN.sol#825)
- _tokenTransfer(from,to,amount,takeFee) (ALDN.sol#1033)

Reentrancy in ALDN.transferFrom(address,address,uint256) (ALDN.sol#702-709):
External calls:
- _transfer(sender,recipient,amount) (ALDN.sol#703)
- overMinTokenBalance && from != owner() && from != address(swapAndLiquify) && ! swapAndLiquify.inSwapAndLiquify() && swapAndLiquifyEnabled (ALDN.sol#1021)
- swapAndLiquify.swapAndLiquify(contractTokenBalance) (ALDN.sol#1024)

Event emitted after the call(s):
- Approval(owner,spender,amount) (ALDN.sol#1005)
- _approve(sender,_msgSender(),spenderAllowance.sub(amount,ERC20: transfer amount exceeds allowance)) (ALDN.sol#706)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ALDN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ALDN.sol#1147-1156) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp <= expiry,ALDN::delegateBySig: signature expired) (ALDN.sol#1154)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (ALDN.sol#23-32) uses assembly
- INLINE ASM (ALDN.sol#30)

Address._verifyCallResult(bool,bytes,string) (ALDN.sol#168-185) uses assembly
- INLINE ASM (ALDN.sol#177-180)

ALDN._getChainId() (ALDN.sol#1318-1322) uses assembly
- INLINE ASM (ALDN.sol#1320)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (ALDN.sol#168-185) is never used and should be removed
Address.functionCall(address,bytes) (ALDN.sol#76-78) is never used and should be removed
Address.functionCall(address,bytes,string) (ALDN.sol#86-88) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ALDN.sol#101-103) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ALDN.sol#111-118) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ALDN.sol#150-152) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ALDN.sol#160-166) is never used and should be removed
Address.functionStaticCall(address,bytes) (ALDN.sol#126-128) is never used and should be removed

Address.functionStaticCall(address,bytes,string) (ALDN.sol#136-142) is never used and should be removed
Address.isContract(address) (ALDN.sol#23-32) is never used and should be removed
Address.sendValue(address,uint256) (ALDN.sol#50-56) is never used and should be removed
Context._msgData() (ALDN.sol#399-402) is never used and should be removed
SafeMath.div(uint256,uint256,string) (ALDN.sol#363-368) is never used and should be removed
SafeMath.mod(uint256,uint256) (ALDN.sol#323-325) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ALDN.sol#385-390) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (ALDN.sol#194-200) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (ALDN.sol#236-241) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (ALDN.sol#248-253) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (ALDN.sol#219-229) is never used and should be removed
SafeMath.trySub(uint256,uint256) (ALDN.sol#207-212) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
ALDN._rTotal (ALDN.sol#582) is set pre-construction with a non-constant function or state variable:
- (MAX - (MAX % _tTotal))
ALDN._previousTaxFee (ALDN.sol#591) is set pre-construction with a non-constant function or state variable:
- _taxFee
ALDN._previousLiquidityFee (ALDN.sol#594) is set pre-construction with a non-constant function or state variable:
- _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.0 (ALDN.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (ALDN.sol#50-56):
- (success) = recipient.call{value: amount}() (ALDN.sol#54)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ALDN.sol#111-118):
- (success,returndata) = target.call{value: value}(data) (ALDN.sol#116)
Low level call in Address.functionStaticCall(address,bytes,string) (ALDN.sol#136-142):
- (success,returndata) = target.staticcall(data) (ALDN.sol#140)
Low level call in Address.functionDelegateCall(address,bytes,string) (ALDN.sol#160-166):
- (success,returndata) = target.delegatecall(data) (ALDN.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ALDN.setSwapAndLiquifyEnabled(bool)._enabled (ALDN.sol#866) is not in mixedCase
Parameter ALDN.calculateTaxFee(uint256)._amount (ALDN.sol#968) is not in mixedCase
Parameter ALDN.calculateLiquidityFee(uint256)._amount (ALDN.sol#972) is not in mixedCase
Variable ALDN._taxFee (ALDN.sol#590) is not in mixedCase
Variable ALDN._liquidityFee (ALDN.sol#593) is not in mixedCase
Variable ALDN._maxTxAmount (ALDN.sol#598) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (ALDN.sol#400)" inContext (ALDN.sol#394-403)
Redundant expression "burnQuantity (ALDN.sol#1126)" inALDN (ALDN.sol#566-1324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable ALDN._delegate(address,address).delegatorROwned (ALDN.sol#1236) is too similar to ALDN._delegate(address,address).delegatorTOwned (ALDN.sol#1236)
Variable ALDN._tokenTransfer(address,address,uint256,bool).newRecipientROwned (ALDN.sol#1068) is too similar to ALDN._tokenTransfer(address,address,uint256,bool).newRecipientTOwned (ALDN.sol#1068)
Variable ALDN._tokenTransfer(address,address,uint256,bool).newSenderROwned (ALDN.sol#1067) is too similar to ALDN._tokenTransfer(address,address,uint256,bool).newSenderTOwned (ALDN.sol#1067)
Variable ALDN._tokenTransfer(address,address,uint256,bool).oldRecipientROwned (ALDN.sol#1043) is too similar to ALDN._tokenTransfer(address,address,uint256,bool).oldRecipientTOwned (ALDN.sol#1043)
Variable ALDN._tokenTransfer(address,address,uint256,bool).oldSenderROwned (ALDN.sol#1042) is too similar to ALDN._tokenTransfer(address,address,uint256,bool).oldSenderTOwned (ALDN.sol#1042)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)

Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._transferToExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1095)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._transferStandard(address,address,uint256).rTransferAmount (ALDN.sol#1076) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN.reflectionFromToken(uint256,bool).rTransferAmount (ALDN.sol#758) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN._transferBothExcluded(address,address,uint256).rTransferAmount (ALDN.sol#812) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._transferBothExcluded(address,address,uint256).tTransferAmount (ALDN.sol#814)
Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)

Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._getTValues(uint256).tTransferAmount (ALDN.sol#887)
Variable ALDN._transferFromExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1111) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._getValues(uint256).rTransferAmount (ALDN.sol#879) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._getValues(uint256).tTransferAmount (ALDN.sol#878)
Variable ALDN._transferToExcluded(address,address,uint256).rTransferAmount (ALDN.sol#1093) is too similar to ALDN._transferStandard(address,address,uint256).tTransferAmount (ALDN.sol#1078)
Variable ALDN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (ALDN.sol#896) is too similar to ALDN._transferFromExcluded(address,address,uint256).tTransferAmount (ALDN.sol#1113)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ALDN.slitherConstructorVariables() (ALDN.sol#566-1324) uses literals with too many digits:
	- _tTotal = 1000000000 * 10 ** 6 * 10 ** 9 (ALDN.sol#581)
ALDN.slitherConstructorVariables() (ALDN.sol#566-1324) uses literals with too many digits:
	- _maxTxAmount = 5000000 * 10 ** 6 * 10 ** 9 (ALDN.sol#598)
ALDN.slitherConstructorVariables() (ALDN.sol#566-1324) uses literals with too many digits:
	- numTokensSellToAddToLiquidity = 500000 * 10 ** 6 * 10 ** 9 (ALDN.sol#599)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ALDN._decimals (ALDN.sol#587) should be constant
ALDN._name (ALDN.sol#585) should be constant
ALDN._symbol (ALDN.sol#586) should be constant
ALDN._tTotal (ALDN.sol#581) should be constant
ALDN.numRateCheckpoints (ALDN.sol#638) should be constant
ALDN.numTokensSellToAddToLiquidity (ALDN.sol#599) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
authorizedNewOwner() should be declared external:
	- Ownable.authorizedNewOwner() (ALDN.sol#439-441)
renounceOwnership(address) should be declared external:
	- Ownable.renounceOwnership(address) (ALDN.sol#475-480)
name() should be declared external:
	- ALDN.name() (ALDN.sol#667-669)
symbol() should be declared external:
	- ALDN.symbol() (ALDN.sol#671-673)
decimals() should be declared external:
	- ALDN.decimals() (ALDN.sol#675-677)
totalSupply() should be declared external:
	- ALDN.totalSupply() (ALDN.sol#679-681)
transfer(address,uint256) should be declared external:
	- ALDN.transfer(address,uint256) (ALDN.sol#688-691)
allowance(address,address) should be declared external:
	- ALDN.allowance(address,address) (ALDN.sol#693-695)
approve(address,uint256) should be declared external:
	- ALDN.approve(address,uint256) (ALDN.sol#697-700)
transferFrom(address,address,uint256) should be declared external:
	- ALDN.transferFrom(address,address,uint256) (ALDN.sol#702-709)
increaseAllowance(address,uint256) should be declared external:
	- ALDN.increaseAllowance(address,uint256) (ALDN.sol#711-714)
decreaseAllowance(address,uint256) should be declared external:
	- ALDN.decreaseAllowance(address,uint256) (ALDN.sol#716-719)
isExcludedFromReward(address) should be declared external:
	- ALDN.isExcludedFromReward(address) (ALDN.sol#721-723)
totalFees() should be declared external:
	- ALDN.totalFees() (ALDN.sol#725-727)

deliver(uint256) should be declared external:
	- ALDN.deliver(uint256) (ALDN.sol#736-750)
reflectionFromToken(uint256,bool) should be declared external:
	- ALDN.reflectionFromToken(uint256,bool) (ALDN.sol#752-761)
excludeFromReward(address) should be declared external:
	- ALDN.excludeFromReward(address) (ALDN.sol#773-787)
excludeFromFee(address) should be declared external:
	- ALDN.excludeFromFee(address) (ALDN.sol#828-830)
includeInFee(address) should be declared external:
	- ALDN.includeInFee(address) (ALDN.sol#832-834)
excludeFromMaxTxAmount(address) should be declared external:
	- ALDN.excludeFromMaxTxAmount(address) (ALDN.sol#836-838)
includeInMaxTxAmount(address) should be declared external:
	- ALDN.includeInMaxTxAmount(address) (ALDN.sol#840-842)
setSwapAndLiquifyAddress(address) should be declared external:
	- ALDN.setSwapAndLiquifyAddress(address) (ALDN.sol#856-864)
setSwapAndLiquifyEnabled(bool) should be declared external:
	- ALDN.setSwapAndLiquifyEnabled(bool) (ALDN.sol#866-870)
isExcludedFromFee(address) should be declared external:
	- ALDN.isExcludedFromFee(address) (ALDN.sol#991-993)
isExcludedFromMaxTxAmount(address) should be declared external:
	- ALDN.isExcludedFromMaxTxAmount(address) (ALDN.sol#995-997)
delegate(address) should be declared external:
	- ALDN.delegate(address) (ALDN.sol#1134-1136)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
	- ALDN.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ALDN.sol#1147-1156)
getPriorVotes(address,uint256) should be declared external:
	- ALDN.getPriorVotes(address,uint256) (ALDN.sol#1199-1232)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ALDN.sol analyzed (7 contracts with 75 detectors), 136 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Slither log >> LPStaking.sol

INFO:Detectors:
LpStaking.stake(uint256,uint128) (LPStaking.sol#744-768) ignores return value by
IERC20(_aliEthPair).transferFrom(_msgSender(),address(this),amount) (LPStaking.sol#750-754)
LpStaking.withdraw(uint256,uint256) (LPStaking.sol#774-810) ignores return value by
IERC20(_aliEthPair).transfer(_msgSender(),amount) (LPStaking.sol#804)
LpStaking.withdraw(uint256,uint256) (LPStaking.sol#774-810) ignores return value by
IALDN(_aliToken).transfer(_msgSender(),available) (LPStaking.sol#807)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LpStaking.getALIPerBlockForReward() (LPStaking.sol#671-682) uses a dangerous strict equality:
	- _totalStakedAmount == 0 (LPStaking.sol#672)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
LpStaking.withdraw(uint256,uint256) (LPStaking.sol#774-810) contains a tautology or contradiction:
	- require(bool,string)(staker.length > index && index >= 0,Stake: Invalid index.) (LPStaking.sol#778)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
LpStaking.getReward(address).i (LPStaking.sol#737) is a local variable never initialized
LpStaking.setMultipliers(uint16[]).i (LPStaking.sol#641) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (LPStaking.sol#457) lacks a zero-check
on :
		- _authorizedNewOwner = authorizedAddress (LPStaking.sol#458)
LpStaking.setTokenAddress(address).tokenAddress (LPStaking.sol#647) lacks a zero-check on :

- _aliToken = tokenAddress (LPStaking.sol#648)
LpStaking.setPairAddressAddress(address).pairAddress (LPStaking.sol#651) lacks a zero-check on :
- _aliEthPair = pairAddress (LPStaking.sol#652)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LpStaking.stake(uint256,uint128) (LPStaking.sol#744-768):
External calls:
- IERC20(_aliEthPair).transferFrom(_msgSender(),address(this),amount) (LPStaking.sol#750-754)
State variables written after the call(s):
- staker.push(StakeInfo(amount,block.timestamp,lockWeek)) (LPStaking.sol#757-763)
- _totalStakedAmount = _totalStakedAmount.add(amount.mul(calcMultiplier(lockWeek)))
(LPStaking.sol#764-766)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LpStaking.stake(uint256,uint128) (LPStaking.sol#744-768):
External calls:
- IERC20(_aliEthPair).transferFrom(_msgSender(),address(this),amount) (LPStaking.sol#750-754)
Event emitted after the call(s):
- Staked(_msgSender(),amount) (LPStaking.sol#767)
Reentrancy in LpStaking.withdraw(uint256,uint256) (LPStaking.sol#774-810):
External calls:
- IERC20(_aliEthPair).transfer(_msgSender(),amount) (LPStaking.sol#804)
- IALDN(_aliToken).transfer(_msgSender(),available) (LPStaking.sol#807)
Event emitted after the call(s):
- Withdraw(_msgSender(),amount,available) (LPStaking.sol#809)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LpStaking.calcMultiplier(uint256) (LPStaking.sol#660-668) uses timestamp for comparisons
Dangerous comparisons:
- numOfWeeks < 4 (LPStaking.sol#661)
- numOfWeeks >= 104 (LPStaking.sol#663)
LpStaking.getALIPerBlockForReward() (LPStaking.sol#671-682) uses timestamp for comparisons
Dangerous comparisons:
- _totalStakedAmount == 0 (LPStaking.sol#672)
LpStaking.getReward(address,uint256) (LPStaking.sol#693-729) uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp >= _startedStakeTime.add(_stakedPeriod) (LPStaking.sol#702)
- getStakedAmount(account,index) <= 0 || stakedPeriod <= 0 (LPStaking.sol#709)
- stakedPeriod > uint256(staker[index].lockWeek).mul(604800) || block.timestamp >=
_startedStakeTime.add(_stakedPeriod) (LPStaking.sol#722-723)
LpStaking.getReward(address) (LPStaking.sol#731-742) uses timestamp for comparisons
Dangerous comparisons:
- i < staker.length (LPStaking.sol#737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (LPStaking.sol#23-32) uses assembly
- INLINE ASM (LPStaking.sol#30)
Address._verifyCallResult(bool,bytes,string) (LPStaking.sol#168-185) uses assembly
- INLINE ASM (LPStaking.sol#177-180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (LPStaking.sol#168-185) is never used and should be removed
Address.functionCall(address,bytes) (LPStaking.sol#76-78) is never used and should be removed
Address.functionCall(address,bytes,string) (LPStaking.sol#86-88) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (LPStaking.sol#101-103) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (LPStaking.sol#111-118) is never used and should be removed
Address.functionDelegateCall(address,bytes) (LPStaking.sol#150-152) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (LPStaking.sol#160-166) is never used and should be removed
Address.functionStaticCall(address,bytes) (LPStaking.sol#126-128) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (LPStaking.sol#136-142) is never used and should be removed

```
Address.sendValue(address,uint256) (LPStaking.sol#50-56) is never used and should be removed
Context._msgData() (LPStaking.sol#405-408) is never used and should be removed
SafeMath.div(uint256,uint256,string) (LPStaking.sol#369-374) is never used and should be removed
SafeMath.mod(uint256,uint256) (LPStaking.sol#329-331) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (LPStaking.sol#391-396) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (LPStaking.sol#346-351) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (LPStaking.sol#200-206) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (LPStaking.sol#242-247) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (LPStaking.sol#254-259) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (LPStaking.sol#225-235) is never used and should be removed
SafeMath.trySub(uint256,uint256) (LPStaking.sol#213-218) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (LPStaking.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (LPStaking.sol#50-56):
        - (success) = recipient.call{value: amount}() (LPStaking.sol#54)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (LPStaking.sol#111-118):
        - (success,returndata) = target.call{value: value}(data) (LPStaking.sol#116)
Low level call in Address.functionStaticCall(address,bytes,string) (LPStaking.sol#136-142):
        - (success,returndata) = target.staticcall(data) (LPStaking.sol#140)
Low level call in Address.functionDelegateCall(address,bytes,string) (LPStaking.sol#160-166):
        - (success,returndata) = target.delegatecall(data) (LPStaking.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable LpStaking._multipliers (LPStaking.sol#617) is not in mixedCase
Variable LpStaking._aliToken (LPStaking.sol#622) is not in mixedCase
Variable LpStaking._aliEthPair (LPStaking.sol#623) is not in mixedCase
Variable LpStaking._stakers (LPStaking.sol#632) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (LPStaking.sol#406)" inContext (LPStaking.sol#400-409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
LpStaking._stakedPeriod (LPStaking.sol#614) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
authorizedNewOwner() should be declared external:
        - Ownable.authorizedNewOwner() (LPStaking.sol#445-447)
renounceOwnership(address) should be declared external:
        - Ownable.renounceOwnership(address) (LPStaking.sol#481-486)
setMultipliers(uint16[]) should be declared external:
        - LpStaking.setMultipliers(uint16[]) (LPStaking.sol#639-645)
setTokenAddress(address) should be declared external:
        - LpStaking.setTokenAddress(address) (LPStaking.sol#647-649)
setPairAddressAddress(address) should be declared external:
        - LpStaking.setPairAddressAddress(address) (LPStaking.sol#651-653)
stake(uint256,uint128) should be declared external:
        - LpStaking.stake(uint256,uint128) (LPStaking.sol#744-768)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LPStaking.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 >> NFTStaking.sol

```
INFO:Detectors:
NFTStaking.withdraw(uint256) (NFTStaking.sol#821-849) ignores return value by
IALDN(_aliToken).transfer(_msgSender(),available) (NFTStaking.sol#846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in NFTStaking.stake(uint256,uint128) (NFTStaking.sol#797-819):
    External calls:
    - IERC721(_magicLamp).transferFrom(_msgSender(),address(this),tokenId) (NFTStaking.sol#809)
    State variables written after the call(s):
    - staker.push(StakeInfo(tokenId,block.timestamp,lockWeek)) (NFTStaking.sol#810-816)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
NFTStaking.withdraw(uint256) (NFTStaking.sol#821-849) contains a tautology or contradiction:
    - require(bool,string)(staker.length > index && index >= 0,Stake: Invalid index.) (NFTStaking.sol#825)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
NFTStaking.getReward(address).i (NFTStaking.sol#790) is a local variable never initialized
NFTStaking.setMultipliers(uint16[]).i (NFTStaking.sol#696) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (NFTStaking.sol#589) lacks a zero-check
on :
        - _authorizedNewOwner = authorizedAddress (NFTStaking.sol#590)
NFTStaking.setMagicLampAddress(address).magicLampAddress (NFTStaking.sol#702) lacks a zero-check
on :
        - _magicLamp = magicLampAddress (NFTStaking.sol#703)
NFTStaking.setTokenAddress(address).tokenAddress (NFTStaking.sol#706) lacks a zero-check on :
        - _aliToken = tokenAddress (NFTStaking.sol#707)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in NFTStaking.stake(uint256,uint128) (NFTStaking.sol#797-819):
    External calls:
    - IERC721(_magicLamp).transferFrom(_msgSender(),address(this),tokenId) (NFTStaking.sol#809)
    State variables written after the call(s):
    - _totalStakedAmount = _totalStakedAmount.add(calcMultiplier(lockWeek)) (NFTStaking.sol#817)
Reentrancy in NFTStaking.withdraw(uint256) (NFTStaking.sol#821-849):
    External calls:
    - IERC721(_magicLamp).approve(_msgSender(),tokenId) (NFTStaking.sol#828)
    - IERC721(_magicLamp).transferFrom(address(this),_msgSender(),tokenId) (NFTStaking.sol#829)
    State variables written after the call(s):
    - _totalStakedAmount = _totalStakedAmount.sub(stakedAmount) (NFTStaking.sol#832)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in NFTStaking.stake(uint256,uint128) (NFTStaking.sol#797-819):
    External calls:
    - IERC721(_magicLamp).transferFrom(_msgSender(),address(this),tokenId) (NFTStaking.sol#809)
    Event emitted after the call(s):
    - Staked(_msgSender(),tokenId) (NFTStaking.sol#818)
Reentrancy in NFTStaking.withdraw(uint256) (NFTStaking.sol#821-849):
    External calls:
    - IERC721(_magicLamp).approve(_msgSender(),tokenId) (NFTStaking.sol#828)
    - IERC721(_magicLamp).transferFrom(address(this),_msgSender(),tokenId) (NFTStaking.sol#829)
    - IALDN(_aliToken).transfer(_msgSender(),available) (NFTStaking.sol#846)
    Event emitted after the call(s):
    - Withdraw(_msgSender(),tokenId,available) (NFTStaking.sol#848)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
NFTStaking.calcMultiplier(uint256) (NFTStaking.sol#714-722) uses timestamp for comparisons
    Dangerous comparisons:
    - numOfWeeks < 4 (NFTStaking.sol#715)
    - numOfWeeks >= 104 (NFTStaking.sol#717)
NFTStaking.getReward(address,uint256) (NFTStaking.sol#746-782) uses timestamp for comparisons
    Dangerous comparisons:
    - block.timestamp >= _startedStakeTime.add(_stakedPeriod) (NFTStaking.sol#755)
```

- getStakedAmount(account,index) <= 0 || stakedPeriod <= 0 (NFTStaking.sol#762)
- stakedPeriod > uint256(staker[index].lockWeek).mul(604800) || block.timestamp >= _startedStakeTime.add(_stakedPeriod) (NFTStaking.sol#775-776)

NFTStaking.getReward(address) (NFTStaking.sol#784-795) uses timestamp for comparisons
Dangerous comparisons:
- i < staker.length (NFTStaking.sol#790)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (NFTStaking.sol#161-170) uses assembly
- INLINE ASM (NFTStaking.sol#168)

Address._verifyCallResult(bool,bytes,string) (NFTStaking.sol#306-323) uses assembly
- INLINE ASM (NFTStaking.sol#315-318)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (NFTStaking.sol#306-323) is never used and should be removed
Address.functionCall(address,bytes) (NFTStaking.sol#214-216) is never used and should be removed
Address.functionCall(address,bytes,string) (NFTStaking.sol#224-226) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (NFTStaking.sol#239-241) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (NFTStaking.sol#249-256) is never used and should be removed
Address.functionDelegateCall(address,bytes) (NFTStaking.sol#288-290) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (NFTStaking.sol#298-304) is never used and should be removed
Address.functionStaticCall(address,bytes) (NFTStaking.sol#264-266) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (NFTStaking.sol#274-280) is never used and should be removed
Address.sendValue(address,uint256) (NFTStaking.sol#188-194) is never used and should be removed
Context._msgData() (NFTStaking.sol#537-540) is never used and should be removed
SafeMath.div(uint256,uint256,string) (NFTStaking.sol#501-506) is never used and should be removed
SafeMath.mod(uint256,uint256) (NFTStaking.sol#461-463) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (NFTStaking.sol#523-528) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (NFTStaking.sol#478-483) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (NFTStaking.sol#332-338) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (NFTStaking.sol#374-379) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (NFTStaking.sol#386-391) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (NFTStaking.sol#357-367) is never used and should be removed
SafeMath.trySub(uint256,uint256) (NFTStaking.sol#345-350) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (NFTStaking.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (NFTStaking.sol#188-194):
- (success) = recipient.call{value: amount}() (NFTStaking.sol#192)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (NFTStaking.sol#249-256):
- (success,returndata) = target.call{value: value}(data) (NFTStaking.sol#254)

Low level call in Address.functionStaticCall(address,bytes,string) (NFTStaking.sol#274-280):
- (success,returndata) = target.staticcall(data) (NFTStaking.sol#278)

Low level call in Address.functionDelegateCall(address,bytes,string) (NFTStaking.sol#298-304):
- (success,returndata) = target.delegatecall(data) (NFTStaking.sol#302)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable NFTStaking._multipliers (NFTStaking.sol#673) is not in mixedCase
Variable NFTStaking._aliToken (NFTStaking.sol#678) is not in mixedCase
Variable NFTStaking._magicLamp (NFTStaking.sol#679) is not in mixedCase
Variable NFTStaking._stakers (NFTStaking.sol#687) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (NFTStaking.sol#538)" inContext (NFTStaking.sol#532-541)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
NFTStaking._stakedPeriod (NFTStaking.sol#670) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
authorizedNewOwner() should be declared external:
- Ownable.authorizedNewOwner() (NFTStaking.sol#577-579)
renounceOwnership(address) should be declared external:
- Ownable.renounceOwnership(address) (NFTStaking.sol#613-618)
setMultipliers(uint16[]) should be declared external:
- NFTStaking.setMultipliers(uint16[]) (NFTStaking.sol#694-700)
setMagicLampAddress(address) should be declared external:
- NFTStaking.setMagicLampAddress(address) (NFTStaking.sol#702-704)
setTokenAddress(address) should be declared external:
- NFTStaking.setTokenAddress(address) (NFTStaking.sol#706-708)
stake(uint256,uint128) should be declared external:
- NFTStaking.stake(uint256,uint128) (NFTStaking.sol#797-819)
withdraw(uint256) should be declared external:
- NFTStaking.withdraw(uint256) (NFTStaking.sol#821-849)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:NFTStaking.sol analyzed (9 contracts with 75 detectors), 56 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Slither log >> SwapAndLiquify.sol

INFO:Detectors:
SwapAndLiquify.swapAndLiquify(uint256) (SwapAndLiquify.sol#598-610) ignores return value by
IERC20(aldnAddress).transferFrom(_msgSender(),address(this),tokenAmount) (SwapAndLiquify.sol#603)
SwapAndLiquify.initializeLiquidity(uint256,uint256) (SwapAndLiquify.sol#627-635) ignores return value by
IERC20(aldnAddress).transferFrom(_msgSender(),address(this),tokenAmount) (SwapAndLiquify.sol#630)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
SwapAndLiquify.constructor(address) (SwapAndLiquify.sol#580-593) ignores return value by
IERC20(aldnAddress).approve(address(_uniswapV2Router),type()(uint256).max) (SwapAndLiquify.sol#592)
SwapAndLiquify.addLiquidity(uint256,uint256) (SwapAndLiquify.sol#637-647) ignores return value by
uniswapV2Router.addLiquidityETH{value:
ethAmount}(aldnAddress,tokenAmount,0,0,address(this),(block.timestamp)) (SwapAndLiquify.sol#639-646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (SwapAndLiquify.sol#343) lacks a
zero-check on :
- _authorizedNewOwner = authorizedAddress (SwapAndLiquify.sol#344)
SwapAndLiquify.constructor(address)._aldnAddress (SwapAndLiquify.sol#580) lacks a zero-check on :
- aldnAddress = _aldnAddress (SwapAndLiquify.sol#581)
SwapAndLiquify.constructor(address)._uniswapV2Pair (SwapAndLiquify.sol#586-587) lacks a zero-check on
:
- uniswapV2Pair = _uniswapV2Pair (SwapAndLiquify.sol#590)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in SwapAndLiquify.constructor(address) (SwapAndLiquify.sol#580-593):
External calls:
- _uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(aldnAddress,_uniswapV2Router.WETH())
(SwapAndLiquify.sol#586-587)
State variables written after the call(s):
- uniswapV2Pair = _uniswapV2Pair (SwapAndLiquify.sol#590)
- uniswapV2Router = _uniswapV2Router (SwapAndLiquify.sol#589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:

Reentrancy in SwapAndLiquify.initializeLiquidity(uint256,uint256) (SwapAndLiquify.sol#627-635):
External calls:
- IERC20(aldnAddress).transferFrom(_msgSender(),address(this),tokenAmount) (SwapAndLiquify.sol#630)
- addLiquidity(tokenAmount,ethAmount) (SwapAndLiquify.sol#632)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(aldnAddress,tokenAmount,0,0,address(this),(block.timestamp)) (SwapAndLiquify.sol#639-646)
External calls sending eth:
- addLiquidity(tokenAmount,ethAmount) (SwapAndLiquify.sol#632)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(aldnAddress,tokenAmount,0,0,address(this),(block.timestamp)) (SwapAndLiquify.sol#639-646)
Event emitted after the call(s):
- LiquidityInitialized(tokenAmount,ethAmount) (SwapAndLiquify.sol#634)
Reentrancy in SwapAndLiquify.swapAndLiquify(uint256) (SwapAndLiquify.sol#598-610):
External calls:
- IERC20(aldnAddress).transferFrom(_msgSender(),address(this),tokenAmount) (SwapAndLiquify.sol#603)
- swapTokensForEth(half) (SwapAndLiquify.sol#605)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (SwapAndLiquify.sol#618-624)
- addLiquidity(otherHalf,spendableBalance) (SwapAndLiquify.sol#607)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(aldnAddress,tokenAmount,0,0,address(this),(block.timestamp)) (SwapAndLiquify.sol#639-646)
External calls sending eth:
- addLiquidity(otherHalf,spendableBalance) (SwapAndLiquify.sol#607)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(aldnAddress,tokenAmount,0,0,address(this),(block.timestamp)) (SwapAndLiquify.sol#639-646)
Event emitted after the call(s):
- SwapAndLiquified(half,spendableBalance,otherHalf) (SwapAndLiquify.sol#609)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (SwapAndLiquify.sol#291-294) is never used and should be removed
SafeMath.add(uint256,uint256) (SwapAndLiquify.sol#158-160) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SwapAndLiquify.sol#256-261) is never used and should be removed
SafeMath.mod(uint256,uint256) (SwapAndLiquify.sol#216-218) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SwapAndLiquify.sol#278-283) is never used and should be removed
SafeMath.mul(uint256,uint256) (SwapAndLiquify.sol#186-188) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (SwapAndLiquify.sol#233-238) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (SwapAndLiquify.sol#87-93) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (SwapAndLiquify.sol#129-134) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (SwapAndLiquify.sol#141-146) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (SwapAndLiquify.sol#112-122) is never used and should be removed
SafeMath.trySub(uint256,uint256) (SwapAndLiquify.sol#100-105) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (SwapAndLiquify.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (SwapAndLiquify.sol#401) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (SwapAndLiquify.sol#402) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (SwapAndLiquify.sol#416) is not in mixedCase
Function IUniswapV2Router01.WETH() (SwapAndLiquify.sol#434) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SwapAndLiquify.sol#292)" inContext (SwapAndLiquify.sol#286-295)
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 (SwapAndLiquify.sol#438) is too similar to

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
tBDesired (SwapAndLiquify.sol#439)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
authorizedNewOwner() should be declared external:
- Ownable.authorizedNewOwner() (SwapAndLiquify.sol#331-333)
renounceOwnership(address) should be declared external:
- Ownable.renounceOwnership(address) (SwapAndLiquify.sol#367-372)
swapAndLiquify(uint256) should be declared external:
- SwapAndLiquify.swapAndLiquify(uint256) (SwapAndLiquify.sol#598-610)
initializeLiquidity(uint256,uint256) should be declared external:
- SwapAndLiquify.initializeLiquidity(uint256,uint256) (SwapAndLiquify.sol#627-635)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SwapAndLiquify.sol analyzed (9 contracts with 75 detectors), 34 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Email: audit@EtherAuthority.io