Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer: Jadeite Token
Website: jadeite.site
Platform: Binance Smart Chain
Language: Solidity
Date: September 15th, 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 THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Jadeite Token team to perform the Security audit of Jadeite Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on September 15th, 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

The Jadeite team analyzed the market and identified real opportunities to fill the gaps present today within the cryptocurrency, especially on Binance Smart Chain BSC. It is run by a team of five, two founding members and three Technical Specialists.

Audit scope

Name	**Code Review and Security Analysis Report for Jadeite Token Smart Contract**
Platform	**BSC / Solidity**
File	JadeiteToken.sol
File MD5 Hash	84FFB5E68177D6E49F85F8F0C8CD8F05
Online Code	https://bscscan.com/address/0xf68e57342e4e52102eb1c0dd5b1afabcca57aaee#code
Audit Date	September 15th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Jadeite • Symbol: JDT • Decimals: 9 • Total supply: 100 Billion JDT • Airdrop Unlocked: 3.6% JDT • Marketing Wallet Locked: 10% JDT • Seed Investment Pool Locked: 6% JDT	**YES, This is valid.**
• Holders Rewards: 3% • LP Pool: 3% • Burn Fees: 3% • Marketing Wallet: 1%	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract contains owner control, which does not make it fully decentralized.

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

We found 0 critical, 0 high, 0 medium and 8 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	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract file. Smart contracts also contain Libraries, Smart contracts, inherits and Interfaces. These are compact and well written contracts.

The libraries in Jadeite Token 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 Jadeite Token.

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

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

Documentation

We were given a Jadeite Token smart contract code in the form of an BscScan web link. The hash of that code is mentioned above in the table.

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

Another source of information was its official website https://jadeite.site which provided rich information about the project architecture and tokenomics.

Use of Dependencies

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

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

(1) Interface

(a) IERC20

(b) IUniswapV2Factory

(c) IUniswapV2Pair

(d) IUniswapV2Router01

(e) IUniswapV2Router02

(2) Inherited contracts

(a) Context

(b) Ownable

(3) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

(4) Events

(a) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);

(b) event SwapAndLiquifyEnabledUpdated(bool enabled);

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

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Wrong Seed Investment Pool address	Refer audit findings section below
2	lockTheSwap	modifier	Passed	No Issue
3	name	read	Passed	No Issue
4	symbol	read	Passed	No Issue
5	decimals	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue

12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	isExcludedFromReward	read	Passed	No Issue
15	totalFees	read	Passed	No Issue
16	totalBurn	read	Passed	No Issue
17	deliver	write	Passed	No Issue
18	reflectionFromToken	read	Passed	No Issue
19	tokenFromReflection	read	Passed	No Issue
20	excludeFromReward	write	access only Owner	No Issue
21	includeInReward	external	Infinite Loop	Refer audit findings section below
22	_transferBothExcluded	write	Passed	No Issue
23	receive	external	Passed	No Issue
24	_reflectFee	write	Passed	No Issue
25	_getValues	read	Passed	No Issue
26	_getTValues	read	Passed	No Issue
27	_getRValues	write	Passed	No Issue
28	_getRate	read	Passed	No Issue
29	_getCurrentSupply	read	Passed	No Issue
30	_takeLiquidity	write	Passed	No Issue
31	calculateTaxFee	read	Passed	No Issue
32	calculateLiquidityFee	read	Passed	No Issue
33	removeAllFee	write	Passed	No Issue
34	restoreAllFee	write	Passed	No Issue
35	isExcludedFromFee	read	Passed	No Issue
36	_approve	write	Passed	No Issue
37	_transfer	write	Passed	No Issue
38	swapAndLiquify	write	Passed	No Issue
40	swapTokensForEth	write	Passed	No Issue
41	addLiquidity	write	Centralized risk in addLiquidity	Refer audit findings section below
42	_tokenTransfer	write	Passed	No Issue
43	_transferStandard	write	Passed	No Issue
44	_transferToExcluded	write	Passed	No Issue
45	_transferFromExcluded	write	Passed	No Issue
46	excludeFromFee	write	access only Owner	No Issue
47	includeInFee	write	access only Owner	No Issue
48	enableAllFees	external	access only Owner	No Issue
49	disableAllFees	external	access only Owner	No Issue

50	setmarketingWallet	external	Missing zero address validation	Refer audit findings section below
51	setRouterAddress	write	access only Owner	No Issue
52	setMaxTxPercent	external	Function input parameters lack of check	Refer audit findings section below
53	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
54	setnumTokensSellToAddTo Liquidity	external	access only Owner	No Issue
55	disableMarketingFee	external	access only Owner	No Issue
56	owner	read	Passed	No Issue
57	onlyOwner	modifier	Passed	No Issue
58	renounceOwnership	write	Possible to gain ownership	Refer audit findings section below
59	transferOwnership	write	access only Owner	No Issue
60	geUnlockTime	read	Passed	No Issue
61	lock	write	access only Owner	No Issue
62	unlock	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
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.

Email: audit@EtherAuthority.io

Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Division before multiplication:

```
uniswapV2Router = _uniswapV2Router;
_rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6); // Seed Investment Pool - 6%
_rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10); // Marketing wallet - 10%
_rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36); // Airdrop - 3.6%
_rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804); // Owner 80.4%
```

Solidity being resource constraint language, dividing any amount and then multiplying will cause discrepancy in the outcome. Therefore always multiply the amount first and then divide it.

Resolution: Line no 790,791,792,793 Consider ordering multiplication before division.

Status: **Acknowledged.**

(2) Centralized risk in addLiquidity:

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0,
        0,
        owner(),
        block.timestamp
    );
}
```

In addLiquidity function, the owner gets JDT Tokens from the Pool. If the private key of the owner would be compromised, then it would create a problem.

Resolution: Ideally this can be a governance smart contract. On the other hand, the owner can accept this risk and handle the private key very securely.

Status: **Acknowledged.**

(3) Missing zero address validation:

```
function setmarketingWallet(address newWallet) external onlyOwner() {
    marketingWallet = newWallet;
}
```

Detects missing zero address validation in setGovernance function.

Resolution: Line no 1236 : Check that the _governance address is not zero.

Status: **Acknowledged.**

(4) Variable could be declared as constant:

```
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal = 100000 * 10**6 * 10**9;
uint256 private _rTotal = (MAX - (MAX % _tTotal));
uint256 private _tFeeTotal;

string private _name = "Jadeite";
string private _symbol = "JDT";
uint8 private _decimals = 9;
```

The State variables that never change need to be declared as constants, Variables _name, symbol , _decimals, _tTotal should be declared as constant..

Resolution: Line no : 744,745,746,740 Declare these Variables as constants : _name, _symbol , _decimals, _tTotal.

Status: **Acknowledged.**

(5) Function input parameters lack of check:

```
function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
    require(maxTxPercent > 1, "Cannot set transaction amount less than 1 percent!");
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(
        10**2
    );
}
```

To set a percentage value, the value should not be greater than 100.

Resolution: We suggest that the condition to check value should not be greater than 100.

Status: **Acknowledged.**

(6) Wrong Seed Investment Pool address to exclude from fee:

```
constructor () public {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    uniswapV2Router = _uniswapV2Router;
    _rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6); // Seed Investment Pool - 6%
    _rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10); // Marketing Wallet - 10%
    _rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36); // Airdrop - 3.6%
    _rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804); // Owner 80.4%

    _isExcludedFromFee[owner()] = true;
    _isExcludedFromFee[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = true; //MMarketing wallet
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Airdrop
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Seed Investment Pool
    _isExcludedFromFee[address(this)] = true;

    emit Transfer(address(0), 0x397f149591E9A35BE0eA47F62693bB01ec0dc030, 6000 * 10**6 * 10**9); // Seed Investment Pool - 6%
    emit Transfer(address(0), 0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9, 10000 * 10**6 * 10**9); // Marketing wallet - 10%
    emit Transfer(address(0), 0x314c634c99B976443968190B64CDcd89C128f783, 3600 * 10**6 * 10**9); // Airdrop - 3.6%
    emit Transfer(address(0), 0x7335d00c721c15f395E406F198A0E30e44b76AA3, 80400 * 10**6 * 10**9); // Owner - 80.4%
}
```

Here Seed Investment Pool address to exclude from fee is same as Airdrop address.

Resolution: We suggest you correct this address.

Status: **Acknowledged.**

(7) Infinite Loop:

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    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;
        }
    }
}
```

The excludeMultipleAccountsFromFees function for loop does not have accounts length limit , which costs more gas.

Resolution: Upper limit should be limited in for loops

Status: **Acknowledged.**

(8) Possible to gain ownership:

Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

1. Owner calls the lock function in contract to set the current owner as _previousOwner.
2. Owner calls unlock to unlock the contract and set _owner = _previousOwner.
3. Owner called renounceOwnership to leave the contract without the owner.
4. Owner calls unlock to regain ownership.

Resolution: Remove these lock/unlock functions as this seems not serving a great purpose OR always renounce ownership first before calling the lock function

Status: **Acknowledged.**

Very Low / Informational / Best practices:

(1) Solidity version:

```
pragma solidity 0.6.12;
```

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

Resolution: Please use 0.8.7 which is the latest version.

Status: **Acknowledged.**

(2) Ignored return value:

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0,
    0,
    owner(),
    block.timestamp
);
```

The addLiquidityETH is not a void-returning function. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when falling.

Resolution: Line no 1084 : We recommend checking the output of addLiquidityETH before continuing processing.

Status: **Acknowledged.**

(3) Uniqueness for owner address :

```
constructor () public {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    uniswapV2Router = _uniswapV2Router;
    _rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6); // Seed Investment Pool - 6%
    _rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10); // Marketing wallet - 10%
    _rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36); // Airdrop - 3.6%
    _rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804); // Owner 80.4%

    _isExcludedFromFee[owner()] = true;
    _isExcludedFromFee[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = true; //MMarketing wallet
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Airdrop
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Seed Investment Pool
    _isExcludedFromFee[address(this)] = true;

    emit Transfer(address(0), 0x397f149591E9A35BE0eA47F62693bB01ec0dc030, 6000 * 10**6 * 10**9); // Seed Investment Pool - 6%
    emit Transfer(address(0), 0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9, 10000 * 10**6 * 10**9); // Marketing wallet - 10%
    emit Transfer(address(0), 0x314c634c99B976443968190B64CDcd89C128f783, 3600 * 10**6 * 10**9); // Airdrop - 3.6%
    emit Transfer(address(0), 0x7335d00c721c15f395E406F198A0E30e44b76AA3, 80400 * 10**6 * 10**9); // Owner - 80.4%
}
```

Here to set % for the owner and in the Transfer event, hardcoded address has been used.

Resolution: We suggest to use Owner() same as used to exclude from fee.

Status: **Acknowledged.**

(4) All functions which are not called internally, must be declared as external. It is more efficient as sometimes it saves some gas.

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

Status: **Acknowledged.**

(5) Use of hardcoded address:

```
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    uniswapV2Router = _uniswapV2Router;
    _rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6); // Seed Investment Pool - 6%
    _rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10); // Marketing wallet - 10%
    _rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36); // Airdrop - 3.6%
    _rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804); // Owner 80.4%

    _isExcludedFromFee[owner()] = true;
    _isExcludedFromFee[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = true; //MMarketing wallet
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Airdrop
    _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true; //Seed Investment Pool
    _isExcludedFromFee[address(this)] = true;

    emit Transfer(address(0), 0x397f149591E9A35BE0eA47F62693bB01ec0dc030, 6000 * 10**6 * 10**9); // Seed Investment Pool - 6%
    emit Transfer(address(0), 0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9, 10000 * 10**6 * 10**9); // Marketing wallet - 10%
    emit Transfer(address(0), 0x314c634c99B976443968190B64CDcd89C128f783, 3600 * 10**6 * 10**9); // Airdrop - 3.6%
    emit Transfer(address(0), 0x7335d00c721c15f395E406F198A0E30e44b76AA3, 80400 * 10**6 * 10**9); // Owner - 80.4%
}

function name() public view returns (string memory) {
    return _name;
```

There is a variable defined in code for marketing wallet. So that variable can be used here to avoid the repetition of the address.

Resolution: We suggest using a defined variable for marketing wallet.

Status: **Acknowledged.**

(6) Remove extra 'M' from comment:

```
_isExcludedFromFee[owner()] = true;
_isExcludedFromFee[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = true; //MMarketing wallet
```

Resolution: correct the spelling from 'MMarketing' to 'Marketing'

Status: **Acknowledged.**

Centralization

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

- excludeFromReward: The Owner can check if the account is already excluded or not.
- includeInReward: The Owner can check if the account is already excluded or not and set the account.
- excludeFromFee: The Owner can set variable: _isExcludedFromFee status true.
- includeInFee: The Owner can set variable: _isExcludedFromFee status false.
- enableAllFees: The Owner can set all fees enabled.
- disableAllFees: The Owner can set all fees disable.
- setmarketingWallet: The Owner can set a new marketing wallet address.
- setRouterAddress: The Owner can set a new router wallet address.
- setMaxTxPercent: The Owner can set a max tax percentage.
- setSwapAndLiquifyEnabled: The Owner can set swap and liquify enable status.
- setnumTokensSellToAddToLiquidity: The Owner can set the number of tokens sold to add to lp.
- disableMarketingFee: The Owner can disable the marketing fee.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are not critical ones. So, **it's good to go to production**.

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

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

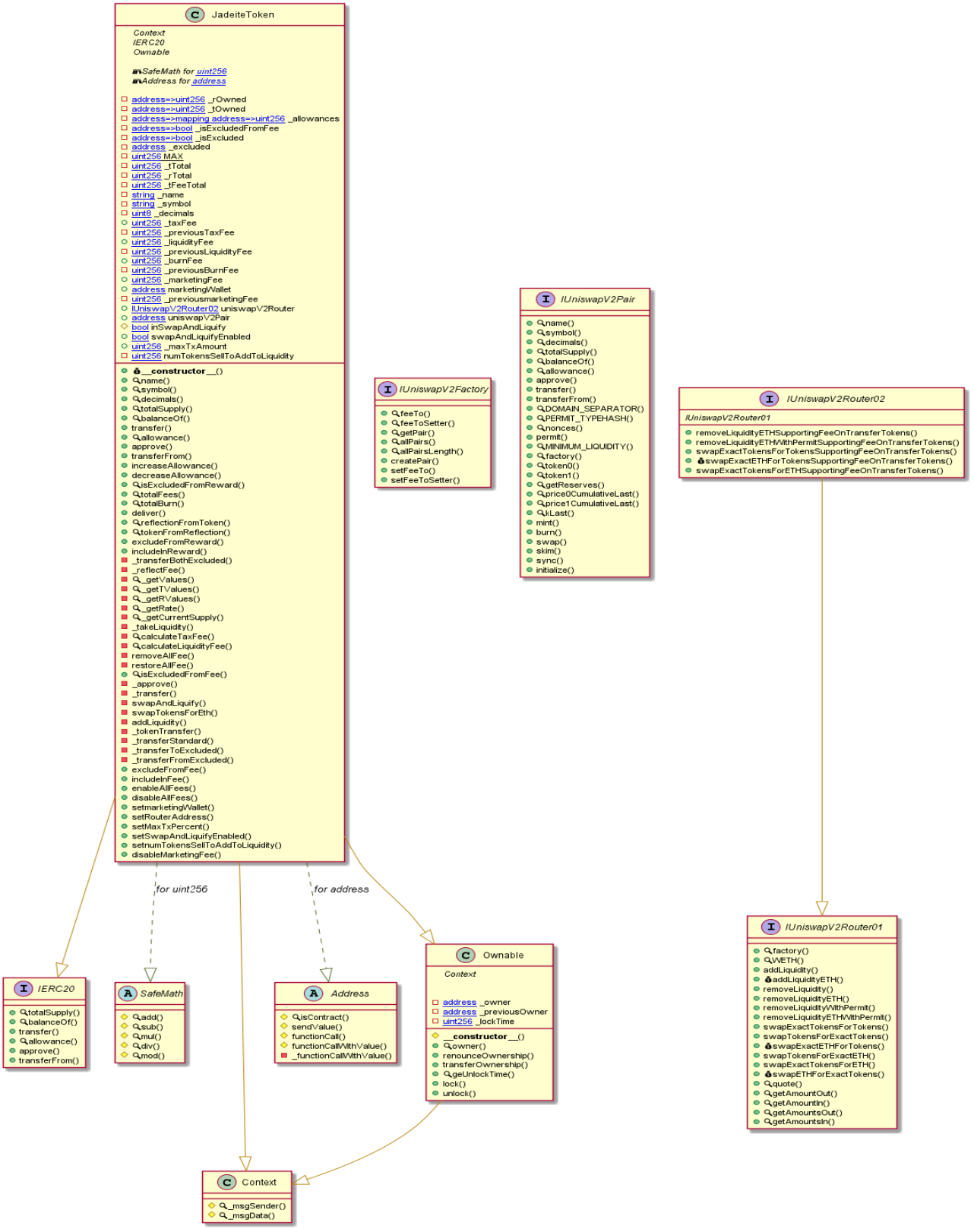
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - Jadeite Token



Slither Results Log

Slither log >> JadeiteToken.sol

```
INFO:Detectors:
JadeiteToken.addLiquidity(uint256,uint256) (JadeiteToken.sol#1031-1042) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.sol#1034
-1041)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in JadeiteToken._transfer(address,address,uint256) (JadeiteToken.sol#976-998):
        External calls:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (JadeiteToken.sol#927)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (JadeiteToken.sol#1086)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (JadeiteToken.sol#1095)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (JadeiteToken.sol#871)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (JadeiteToken.sol#1087)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (JadeiteToken.sol#1106)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (JadeiteToken.sol#1097)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (JadeiteToken.sol#1107)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (JadeiteToken.sol#873)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _rTotal = _rTotal.sub(rFee) (JadeiteToken.sol#882)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (JadeiteToken.sol#929)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (JadeiteToken.sol#870)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (JadeiteToken.sol#1105)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (JadeiteToken.sol#1096)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (JadeiteToken.sol#872)
```

```
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (JadeiteToken.sol#872)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
JadeiteToken.constructor() (JadeiteToken.sol#734-755) performs a multiplication on the result of a division:
        -_rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6) (JadeiteToken.sol#740)
JadeiteToken.constructor() (JadeiteToken.sol#734-755) performs a multiplication on the result of a division:
        -_rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10) (JadeiteToken.sol#741)
JadeiteToken.constructor() (JadeiteToken.sol#734-755) performs a multiplication on the result of a division:
        -_rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36) (JadeiteToken.sol#742)
JadeiteToken.constructor() (JadeiteToken.sol#734-755) performs a multiplication on the result of a division:
        -_rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804) (JadeiteToken.sol#743)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
JadeiteToken.addLiquidity(uint256,uint256) (JadeiteToken.sol#1031-1042) ignores return value by uniswapV2Router.addLiquidityETH{value: et
hAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.sol#1034-1041)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
JadeiteToken.allowance(address,address).owner (JadeiteToken.sol#783) shadows:
        - Ownable.owner() (JadeiteToken.sol#414-416) (function)
JadeiteToken._approve(address,address,uint256).owner (JadeiteToken.sol#968) shadows:
        - Ownable.owner() (JadeiteToken.sol#414-416) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
JadeiteToken.setmarketingWallet(address).newWallet (JadeiteToken.sol#1149) lacks a zero-check on :
                - marketingWallet = newWallet (JadeiteToken.sol#1150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in JadeiteToken._transfer(address,address,uint256) (JadeiteToken.sol#976-998):
        External calls:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
```

```
sol#1034-1041)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _burnFee = _previousBurnFee (JadeiteToken.sol#960)
                - _burnFee = 0 (JadeiteToken.sol#953)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _liquidityFee = _previousLiquidityFee (JadeiteToken.sol#959)
                - _liquidityFee = 0 (JadeiteToken.sol#952)
                - _liquidityFee = 0 (JadeiteToken.sol#1068)
                - _liquidityFee = _previousLiquidityFee (JadeiteToken.sol#1077)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _marketingFee = _previousmarketingFee (JadeiteToken.sol#961)
                - _marketingFee = 0 (JadeiteToken.sol#954)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _previousBurnFee = _burnFee (JadeiteToken.sol#948)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _previousLiquidityFee = _liquidityFee (JadeiteToken.sol#947)
```

```
                - _previousLiquidityFee = _liquidityFee (JadeiteToken.sol#947)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _previousTaxFee = _taxFee (JadeiteToken.sol#946)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _previousmarketingFee = _marketingFee (JadeiteToken.sol#949)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _tFeeTotal = _tFeeTotal.add(tFee) (JadeiteToken.sol#883)
        - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
                - _taxFee = _previousTaxFee (JadeiteToken.sol#958)
                - _taxFee = 0 (JadeiteToken.sol#951)
                - _taxFee = 0 (JadeiteToken.sol#1067)
                - _taxFee = _previousTaxFee (JadeiteToken.sol#1076)
Reentrancy in JadeiteToken.constructor() (JadeiteToken.sol#734-755):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (JadeiteToken.s
ol#736-737)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (JadeiteToken.sol#745)
        - _isExcludedFromFee[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = true (JadeiteToken.sol#746)
        - _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true (JadeiteToken.sol#747)
        - _isExcludedFromFee[0x314c634c99B976443968190B64CDcd89C128f783] = true (JadeiteToken.sol#748)
        - _isExcludedFromFee[address(this)] = true (JadeiteToken.sol#749)
        - _rOwned[0x397f149591E9A35BE0eA47F62693bB01ec0dc030] = _rTotal.div(100).mul(6) (JadeiteToken.sol#740)
        - _rOwned[0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9] = _rTotal.div(100).mul(10) (JadeiteToken.sol#741)
        - _rOwned[0x314c634c99B976443968190B64CDcd89C128f783] = _rTotal.div(1000).mul(36) (JadeiteToken.sol#742)
        - _rOwned[0x7335d00c721c15f395E406F198A0E30e44b76AA3] = _rTotal.div(1000).mul(804) (JadeiteToken.sol#743)
        - uniswapV2Router = _uniswapV2Router (JadeiteToken.sol#739)
Reentrancy in JadeiteToken.setRouterAddress(address) (JadeiteToken.sol#1153-1157):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory()).createPair(address(this),_newPancakeRouter.WETH()) (JadeiteToken
.sol#1155)
        State variables written after the call(s):
        - uniswapV2Router = _newPancakeRouter (JadeiteToken.sol#1156)
Reentrancy in JadeiteToken.swapAndLiquify(uint256) (JadeiteToken.sol#1000-1013):
        External calls:
        - swapTokensForEth(half) (JadeiteToken.sol#1006)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
```

```
adeiteToken.sol#1022-1028)
        - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
                - _allowances[owner][spender] = amount (JadeiteToken.sol#972)
Reentrancy in JadeiteToken.transferFrom(address,address,uint256) (JadeiteToken.sol#792-796):
        External calls:
        - _transfer(sender,recipient,amount) (JadeiteToken.sol#793)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (JadeiteToken.sol#793)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (JadeiteTo
ken.sol#794)
                - _allowances[owner][spender] = amount (JadeiteToken.sol#972)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in JadeiteToken._transfer(address,address,uint256) (JadeiteToken.sol#976-998):
        External calls:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (JadeiteToken.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
```

```
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (JadeiteToken.sol#1090)
                - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
        - Transfer(sender,recipient,tTransferAmount) (JadeiteToken.sol#1100)
                - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
        - Transfer(sender,recipient,tTransferAmount) (JadeiteToken.sol#1110)
                - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
        - Transfer(sender,recipient,tTransferAmount) (JadeiteToken.sol#876)
                - _tokenTransfer(from,to,amount) (JadeiteToken.sol#997)
Reentrancy in JadeiteToken.constructor() (JadeiteToken.sol#734-755):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (JadeiteToken.s
ol#736-737)
        Event emitted after the call(s):
        - Transfer(address(0),0x397f149591E9A35BE0eA47F62693bB01ec0dc030,6000 * 10 ** 6 * 10 ** 9) (JadeiteToken.sol#751)
        - Transfer(address(0),0x27fF92eF073Fc7788760C39CBF5313bCd80a56F9,10000 * 10 ** 6 * 10 ** 9) (JadeiteToken.sol#752)
        - Transfer(address(0),0x314c634c99B976443968190B64CDcd89C128f783,3600 * 10 ** 6 * 10 ** 9) (JadeiteToken.sol#753)
        - Transfer(address(0),0x7335d00c721c15f395E406F198A0E30e44b76AA3,80400 * 10 ** 6 * 10 ** 9) (JadeiteToken.sol#754)
Reentrancy in JadeiteToken.swapAndLiquify(uint256) (JadeiteToken.sol#1000-1013):
        External calls:
        - swapTokensForEth(half) (JadeiteToken.sol#1006)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
```

```
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (JadeiteToken.sol#973)
                - addLiquidity(otherHalf,newBalance) (JadeiteToken.sol#1010)
        - SwapAndLiquify(half,newBalance,otherHalf) (JadeiteToken.sol#1012)
Reentrancy in JadeiteToken.transferFrom(address,address,uint256) (JadeiteToken.sol#792-796):
        External calls:
        - _transfer(sender,recipient,amount) (JadeiteToken.sol#793)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (J
adeiteToken.sol#1022-1028)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (JadeiteToken.sol#793)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (JadeiteToken.
sol#1034-1041)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (JadeiteToken.sol#973)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (J
adeiteToken.sol#794)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (JadeiteToken.sol#461-466) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked until 7 days) (JadeiteToken.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (JadeiteToken.sol#266-275) uses assembly
        - INLINE ASM (JadeiteToken.sol#273)
Address._functionCallWithValue(address,bytes,uint256,string) (JadeiteToken.sol#359-380) uses assembly
        - INLINE ASM (JadeiteToken.sol#372-375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (JadeiteToken.sol#359-380) is never used and should be removed
Address.functionCall(address,bytes) (JadeiteToken.sol#319-321) is never used and should be removed
Address.functionCall(address,bytes,string) (JadeiteToken.sol#329-331) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (JadeiteToken.sol#344-346) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (JadeiteToken.sol#354-357) is never used and should be removed
Address.isContract(address) (JadeiteToken.sol#266-275) is never used and should be removed
```

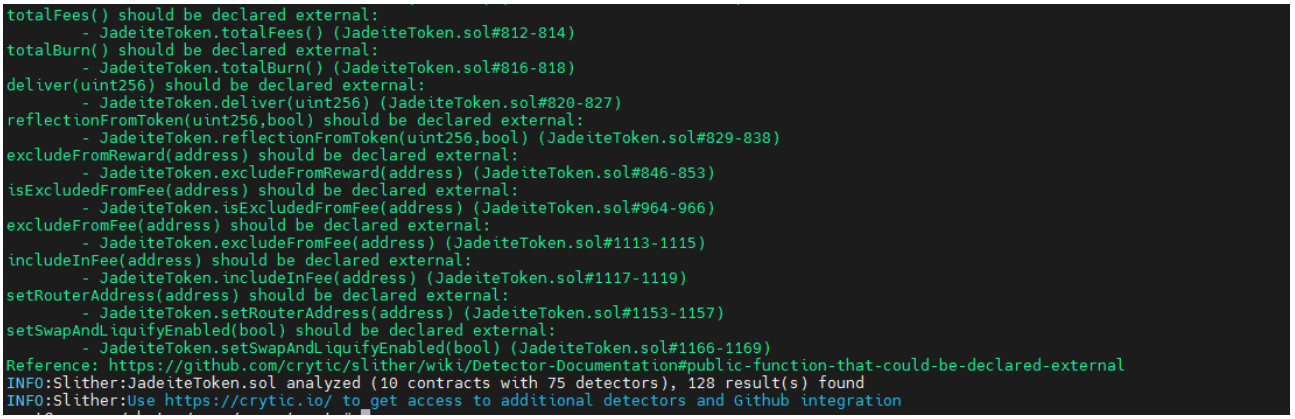
```
Address.isContract(address) (JadeiteToken.sol#266-275) is never used and should be removed
Address.sendValue(address,uint256) (JadeiteToken.sol#293-299) is never used and should be removed
Context._msgData() (JadeiteToken.sol#238-241) is never used and should be removed
SafeMath.mod(uint256,uint256) (JadeiteToken.sol#211-213) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (JadeiteToken.sol#227-230) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
JadeiteToken._rTotal (JadeiteToken.sol#695) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
JadeiteToken._previousTaxFee (JadeiteToken.sol#703) is set pre-construction with a non-constant function or state variable:
        - _taxFee
JadeiteToken._previousLiquidityFee (JadeiteToken.sol#706) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
JadeiteToken._previousBurnFee (JadeiteToken.sol#709) is set pre-construction with a non-constant function or state variable:
        - _burnFee
JadeiteToken._previousmarketingFee (JadeiteToken.sol#713) is set pre-construction with a non-constant function or state variable:
        - _marketingFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (JadeiteToken.sol#293-299):
        - (success) = recipient.call{value: amount}() (JadeiteToken.sol#297)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (JadeiteToken.sol#359-380):
        - (success,returndata) = target.call{value: weiValue}(data) (JadeiteToken.sol#363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (JadeiteToken.sol#505) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (JadeiteToken.sol#506) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (JadeiteToken.sol#523) is not in mixedCase
Function IUniswapV2Router01.WETH() (JadeiteToken.sol#545) is not in mixedCase
Parameter JadeiteToken.calculateTaxFee(uint256)._amount (JadeiteToken.sol#932) is not in mixedCase
Parameter JadeiteToken.calculateLiquidityFee(uint256)._amount (JadeiteToken.sol#938) is not in mixedCase
Parameter JadeiteToken.setSwapAndLiquifyEnabled(bool)._enabled (JadeiteToken.sol#1166) is not in mixedCase
Parameter JadeiteToken.setnumTokensSellToAddToLiquidity(uint256)._numTokensSellToAddToLiquidity (JadeiteToken.sol#1171) is not in mixedCa
se
Variable JadeiteToken._taxFee (JadeiteToken.sol#702) is not in mixedCase
Variable JadeiteToken._liquidityFee (JadeiteToken.sol#705) is not in mixedCase
Variable JadeiteToken._burnFee (JadeiteToken.sol#708) is not in mixedCase
Variable JadeiteToken._marketingFee (JadeiteToken.sol#711) is not in mixedCase
```

```
Variable JadeiteToken._maxTxAmount (JadeiteToken.sol#721) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (JadeiteToken.sol#239)" inContext (JadeiteToken.sol#233-242)
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 (JadeiteToken.so
l#550) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
(JadeiteToken.sol#551)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_transferStandard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._transferStandard(address
,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._transferStandard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._transferToExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._transferToExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_transferToExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_transferBothExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._transferFromExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._transferFromExcluded(add
ress,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._transferBothExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._transferSt
andard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._transferBothExcluded(add
ress,address,uint256).tTransferAmount (JadeiteToken.sol#869)
```

```
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._transferBothExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._transferFromExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_transferFromExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
transferStandard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._transferToExcluded(addre
ss,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._transferToExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._transferStandard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._transferStandard(address,address,uint256).tTransferAmount (JadeiteToken.sol#1085)
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
getValues(uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
getTValues(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._getValues(
uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._getTValues(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._transferBothExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._getValues(uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._transferBo
thExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._transferFr
omExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._getValues(uint256).tTran
sferAmount (JadeiteToken.sol#887)
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._getValues(uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._transferFromExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
```

```
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
transferToExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._transferTo
Excluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1094)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_getTValues(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
transferFromExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#1104)
Variable JadeiteToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (JadeiteToken.sol#903) is too similar to JadeiteToken.
_getValues(uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken.reflectionFromToken(uint256,bool).rTransferAmount (JadeiteToken.sol#835) is too similar to JadeiteToken._getTValues
(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferBothExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#869) is too similar to JadeiteToke
n._getValues(uint256).tTransferAmount (JadeiteToken.sol#887)
Variable JadeiteToken._getValues(uint256).rTransferAmount (JadeiteToken.sol#888) is too similar to JadeiteToken._getTValues(uint256).tTra
nsferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferFromExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1104) is too similar to JadeiteTok
en._getTValues(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferToExcluded(address,address,uint256).rTransferAmount (JadeiteToken.sol#1094) is too similar to JadeiteToken
._getTValues(uint256).tTransferAmount (JadeiteToken.sol#895)
Variable JadeiteToken._transferStandard(address,address,uint256).rTransferAmount (JadeiteToken.sol#1085) is too similar to JadeiteToken._
transferBothExcluded(address,address,uint256).tTransferAmount (JadeiteToken.sol#869)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
JadeiteToken.disableAllFees() (JadeiteToken.sol#1135-1147) uses literals with too many digits:
        - _maxTxAmount = 100000 * 10 ** 6 * 10 ** 9 (JadeiteToken.sol#1144)
JadeiteToken.slitherConstructorVariables() (JadeiteToken.sol#680-1182) uses literals with too many digits:
        - _tTotal = 100000 * 10 ** 6 * 10 ** 9 (JadeiteToken.sol#694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
JadeiteToken._decimals (JadeiteToken.sol#700) should be constant
JadeiteToken._name (JadeiteToken.sol#698) should be constant
JadeiteToken._symbol (JadeiteToken.sol#699) should be constant
JadeiteToken._tTotal (JadeiteToken.sol#694) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (JadeiteToken.sol#433-436)
```

```
        - Ownable.renounceOwnership() (JadeiteToken.sol#433-436)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (JadeiteToken.sol#442-446)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (JadeiteToken.sol#448-450)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (JadeiteToken.sol#453-458)
unlock() should be declared external:
        - Ownable.unlock() (JadeiteToken.sol#461-466)
name() should be declared external:
        - JadeiteToken.name() (JadeiteToken.sol#757-759)
symbol() should be declared external:
        - JadeiteToken.symbol() (JadeiteToken.sol#761-763)
decimals() should be declared external:
        - JadeiteToken.decimals() (JadeiteToken.sol#765-767)
totalSupply() should be declared external:
        - JadeiteToken.totalSupply() (JadeiteToken.sol#769-771)
transfer(address,uint256) should be declared external:
        - JadeiteToken.transfer(address,uint256) (JadeiteToken.sol#778-781)
allowance(address,address) should be declared external:
        - JadeiteToken.allowance(address,address) (JadeiteToken.sol#783-785)
approve(address,uint256) should be declared external:
        - JadeiteToken.approve(address,uint256) (JadeiteToken.sol#787-790)
transferFrom(address,address,uint256) should be declared external:
        - JadeiteToken.transferFrom(address,address,uint256) (JadeiteToken.sol#792-796)
increaseAllowance(address,uint256) should be declared external:
        - JadeiteToken.increaseAllowance(address,uint256) (JadeiteToken.sol#798-801)
decreaseAllowance(address,uint256) should be declared external:
        - JadeiteToken.decreaseAllowance(address,uint256) (JadeiteToken.sol#803-806)
isExcludedFromReward(address) should be declared external:
        - JadeiteToken.isExcludedFromReward(address) (JadeiteToken.sol#808-810)
```



Solidity Static Analysis

JadeiteToken.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 360:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JadeiteToken.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 735:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JadeiteToken.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1016:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JadeiteToken.setRouterAddress(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1154:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 274:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 373:16:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 457:20:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 464:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1028:12:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1041:12:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 298:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 364:50:

Gas & Economy

Gas costs:

Gas requirement of function JadeiteToken.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 443:4:

Gas costs:

Gas requirement of function JadeiteToken.unlock is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 462:4:

Gas costs:

Gas requirement of function JadeiteToken.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 758:4:

Gas costs:

Gas requirement of function JadeiteToken.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 762:4:

Gas costs:

Gas requirement of function JadeiteToken.balanceOf is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 774:4:

Gas costs:

Gas requirement of function JadeiteToken.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 779:4:

Gas costs:

Gas requirement of function JadeiteToken.approve is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 788:4:

Gas costs:

Gas requirement of function JadeiteToken.transferFrom is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 793:4:

Gas costs:

Gas requirement of function JadeiteToken.increaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 799:4:

Gas costs:

Gas requirement of function JadeiteToken.decreaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 804:4:

Gas costs:

Gas requirement of function JadeiteToken.totalBurn is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 817:4:

Gas costs:

Gas requirement of function JadeiteToken.deliver is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 821:4:

Gas costs:

Gas requirement of function JadeiteToken.reflectionFromToken is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 830:4:

Gas costs:

Gas requirement of function JadeiteToken.tokenFromReflection is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 841:4:

Gas costs:

Gas requirement of function JadeiteToken.excludeFromReward is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 847:4:

Gas costs:

Gas requirement of function JadeiteToken.includeInReward is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 856:4:

Gas costs:

Gas requirement of function JadeiteToken.setRouterAddress is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1154:4:

Gas costs:

Gas requirement of function JadeiteToken.setMaxTxPercent is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1160:3:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 858:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 916:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 497:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 119:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 176:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 212:4:

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 267:4:

Constant/View/Pure functions:

JadeiteToken.reflectionFromToken(uint256,bool) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 830:4:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 741:8:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 742:8:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 743:8:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 744:8:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 741:62:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 742:62:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 743:62:

Similar variable names:

JadeiteToken.() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 744:62:

Similar variable names:

JadeiteToken.totalSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 771:15:

Similar variable names:

JadeiteToken.balanceOf(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 775:41:

Similar variable names:

JadeiteToken.balanceOf(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 776:35:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 825:8:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 825:26:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 826:8:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 826:18:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 824:9:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 824:44:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 825:46:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 826:30:

Similar variable names:

JadeiteToken.deliver(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 827:36:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 831:27:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 831:16:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 833:13:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 833:48:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 834:19:

Similar variable names:

JadeiteToken.reflectionFromToken(uint256,bool) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 836:56:

Similar variable names:

JadeiteToken.tokenFromReflection(uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 842:27:

Similar variable names:

JadeiteToken.excludeFromReward(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 849:11:

Similar variable names:

JadeiteToken.excludeFromReward(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 850:12:

Similar variable names:

JadeiteToken.excludeFromReward(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 850:51:

Similar variable names:

JadeiteToken.includeInReward(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 861:16:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 871:8:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 871:26:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 872:8:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 872:26:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 873:8:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 873:29:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 874:8:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 874:29:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:9:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:137:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 871:46:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 872:46:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:26:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:65:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 873:52:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 874:52:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 877:41:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:51:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 870:90:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 876:20:

Similar variable names:

JadeiteToken._transferBothExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 876:26:

Similar variable names:

JadeiteToken._reflectFee(uint256,uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 883:8:

Similar variable names:

JadeiteToken._reflectFee(uint256,uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 883:18:

Similar variable names:

JadeiteToken._reflectFee(uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 883:30:

Similar variable names:

JadeiteToken._reflectFee(uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 884:36:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tTransferAmount" and "rTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 888:9:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tTransferAmount" and "rTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 889:26:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tTransferAmount" and "rTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 890:25:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tTransferAmount" and "rTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 890:48:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tFee" and "rFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 888:34:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tFee" and "rFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 889:51:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tFee" and "rFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 889:88:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tFee" and "rFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 890:42:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "tFee" and "rFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 890:65:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 888:82:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 889:9:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 889:79:

Similar variable names:

JadeiteToken._getValues(uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 890:16:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 901:8:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 901:26:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 904:34:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 905:16:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 902:8:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 902:23:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 904:46:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 905:42:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 903:8:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 903:29:

Similar variable names:

JadeiteToken._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 904:56:

Similar variable names:

JadeiteToken._getRate() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 909:9:

Similar variable names:

JadeiteToken._getRate() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 909:26:

Similar variable names:

JadeiteToken._getRate() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 910:15:

Similar variable names:

JadeiteToken._getRate() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 910:27:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:16:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:51:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 918:34:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 919:34:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 914:26:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 915:26:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:92:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:101:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 921:22:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 921:34:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 921:52:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 921:61:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 914:8:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 915:8:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:40:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 917:75:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 918:12:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 918:22:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 919:12:

Similar variable names:
JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 919:22:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 921:12:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 922:16:

Similar variable names:

JadeiteToken._getCurrentSupply() : Variables have very similar names "rSupply" and "tSupply". Note: Modifiers are currently not considered by this static analysis.
Pos: 922:25:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 928:8:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 928:33:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 930:12:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 930:37:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 927:8:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 927:29:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 928:60:

Similar variable names:

JadeiteToken._takeLiquidity(uint256) : Variables have very similar names "rLiquidity" and "tLiquidity". Note: Modifiers are currently not considered by this static analysis.
Pos: 930:64:

Similar variable names:

JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1087:8:

Similar variable names:

JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1087:26:

Similar variable names:

JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1088:8:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1088:29:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:9:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:137:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1087:46:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:26:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:65:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1088:52:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1091:41:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:51:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1086:90:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1090:20:

Similar variable names:
JadeiteToken._transferStandard(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1090:26:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1096:8:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1096:26:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1097:8:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1097:29:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1098:8:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1098:29:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:9:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:137:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1096:46:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:26:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:65:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1097:52:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1098:52:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1101:41:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:51:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1095:90:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1100:20:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1100:26:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1100:20:

Similar variable names:
JadeiteToken._transferToExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1100:26:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1106:8:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1106:26:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1107:8:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1107:26:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1108:8:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 1108:29:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:9:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:137:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1106:46:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rAmount" and "tAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1107:46:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:26:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:65:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1108:52:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rTransferAmount" and "tTransferAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1111:41:

Similar variable names:
JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:51:

Similar variable names:

JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1105:90:

Similar variable names:

JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1110:20:

Similar variable names:

JadeiteToken._transferFromExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1110:26:

Similar variable names:

JadeiteToken.setMaxTxPercent(uint256) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 1162:23:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 104:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 134:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 159:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 193:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 229:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 295:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 299:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 356:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 361:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 423:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 444:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 463:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 464:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 823:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 831:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 842:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 848:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 857:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 970:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 971:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 982:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 983:8:

Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 984:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1050:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1161:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 159:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 194:20:

Solhint Linter

JadeiteToken.sol

```
JadeiteToken.sol:2:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
JadeiteToken.sol:457:21: Error: Avoid to make time-based decisions in
your business logic
JadeiteToken.sol:464:17: Error: Avoid to make time-based decisions in
your business logic
JadeiteToken.sol:506:5: Error: Function name must be in mixedCase
JadeiteToken.sol:507:5: Error: Function name must be in mixedCase
JadeiteToken.sol:524:5: Error: Function name must be in mixedCase
JadeiteToken.sol:546:5: Error: Function name must be in mixedCase
JadeiteToken.sol:681:1: Error: Contract has 27 states declarations
but allowed no more than 15
JadeiteToken.sol:719:5: Error: Explicitly mark visibility of state
JadeiteToken.sol:880:32: Error: Code contains empty blocks
JadeiteToken.sol:1028:13: Error: Avoid to make time-based decisions
in your business logic
JadeiteToken.sol:1041:13: Error: Avoid to make time-based decisions
in your business logic
```

Software analysis result:

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

Email: audit@EtherAuthority.io