Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Lido Dao
Website:	stake. lido.fi
Platform:	Ethereum
Language:	Solidity
Date:	April 14th, 2024

Table of contents

`

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the smart contracts of Lido Dao Token from stake.lido.fi were audited. 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 April 14th, 2024.

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 provided Solidity code is for the MiniMeToken contract, originally authored by Jordi Baylina. The MiniMeToken is designed to be a versatile and cloneable ERC20 token that allows for decentralized upgrades and governance.
- The MiniMeToken contract provides a powerful tool for creating and managing ERC20 tokens with advanced governance and upgrade capabilities. Its checkpointing system and cloneable nature make it particularly suited for decentralized applications requiring flexible token management and historical state queries.
- The MiniMeToken Contract aims to facilitate easy cloning of the token through token distribution at a specific block, enabling decentralized feature upgrades for DAOs and DApps without affecting the original token.
- In the actual token contract, the default controller is the owner that deploys the contract, so usually, this token will be deployed by a token controller contract, which Giveth will call a "campaign."

Audit scope

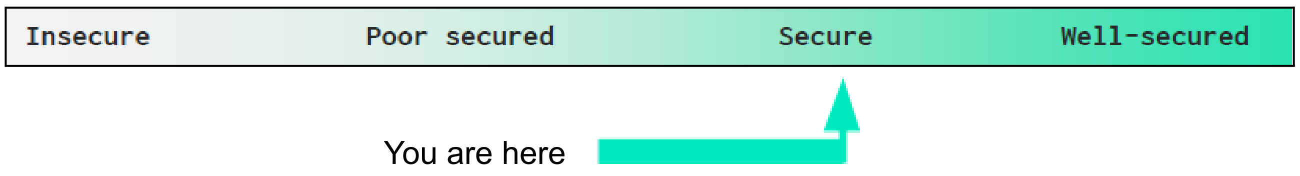
Name	**Code Review and Security Analysis Report for Lido Dao Smart Contract**
Platform	**Ethereum**
File	MiniMeToken.sol
Smart Contract Code	0x5a98fcbea516cf06857215779fd812ca3bef1b32
Audit Date	April 14th, 2024

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Lido DAO Token • Symbol: LDO • Decimals: 18 • Version: MMT_0.1 • Total Supply: 1 billion LDO	**YES, This is valid.**
Security Considerations: • **Controller Authority:** The controller has significant control over the token, including the ability to transfer tokens, mint new tokens, and burn tokens. It is crucial to ensure that the controller is trustworthy and/or governed by a robust DAO mechanism. • **Transfer Restrictions:** Ensure the logic for enabling/disabling transfers is thoroughly tested to prevent unintentional locking or unlocking of tokens.	**YES, This is valid.**
Controller Ownership control: • Generates `_amount` tokens. • Burns `_amount` tokens from `_owner`. • Enables token holders to transfer their tokens freely if true _transfersEnabled True if transfers are allowed in the clone. • extract mistakenly sent tokens to this contract. • Changes the controller of the contract.	**YES, This is valid. We suggest renouncing ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

Audit Summary

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **"Secured"**. Also, these contracts contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhint, and Remix IDE. At the same time, this finding is based on a 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. A general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 1 medium 2 low, and 6 very low-level issues.

Investors Advice: A 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 is not specified	Passed
	Solidity version is too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	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	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Business Risk Analysis

Category	Result
🟢 Buy Tax	0%
🟢 Sell Tax	0%
🟢 Cannot Buy	No
🟢 Cannot Sell	No
🟢 Max Tax	0%
🟢 Modify Tax	Not Detected
🟢 Fee Check	No
🟢 Is Honeypot	Not Detected
🟢 Trading Cooldown	Not Detected
🟢 Can Pause Trade?	No
🟢 Pause Transfer?	Yes
🟢 Max Tax?	No
🟢 Is it Anti-whale?	Yes
🟢 Is Anti-bot?	Not Detected
🟢 Is it a Blacklist?	Not Detected
🟢 Blacklist Check	No
🟢 Can Mint?	No
🟢 Is it Proxy?	No
🟢 Can Take Ownership?	Yes
🟢 Hidden Owner?	No
🟢 Self Destruction?	No
🟢 Auditor Confidence	High

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Lido Dao 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 Lido Dao.

The EtherAuthority team has no scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a Lido Dao Token smart contract code in the form of an Etherscan web link.

As mentioned above, code parts are well commented on. and the logic is straightforward. So it is 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.

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.

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

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createCloneToken	write	Optimization	Refer Audit Findings
3	MiniMeToken	write	Passed	No Issue
4	transfer	write	Passed	No Issue
5	transferFrom	write	Passed	No Issue
6	doTransfer	internal	Passed	No Issue
7	balanceOf	write	Warning	Refer Audit Findings
8	approve	write	Passed	No Issue
9	allowance	write	Passed	No Issue
10	approveAndCall	write	Optimization	Refer Audit Findings
11	totalSupply	write	Passed	No Issue
12	balanceOfAt	write	Passed	No Issue
13	totalSupplyAt	write	Passed	No Issue
14	createCloneToken	write	Optimization	Refer Audit Findings
15	generateTokens	write	Centralization	Refer Audit Findings
16	destroyTokens	write	Centralization	Refer Audit Findings
17	enableTransfers	write	Centralization	Refer Audit Findings
18	getValueAt	internal	Passed	No Issue
19	updateValueAtNow	internal	Passed	No Issue
20	isContract	internal	Passed	No Issue
21	min	internal	Passed	No Issue
22	claimTokens	write	Unchecked-transfer, Warning, Centralization	Refer Audit Findings
23	receiveApproval	write	Optimization	Refer Audit Findings
24	onlyController	modifier	Error message for require	Refer Audit Findings
25	Controlled	write	Passed	No Issue
26	changeController	write	Missing-events-access -control, Missing-zero-address-validation, Centralization	Refer Audit Findings

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.

Audit Findings

Critical Severity

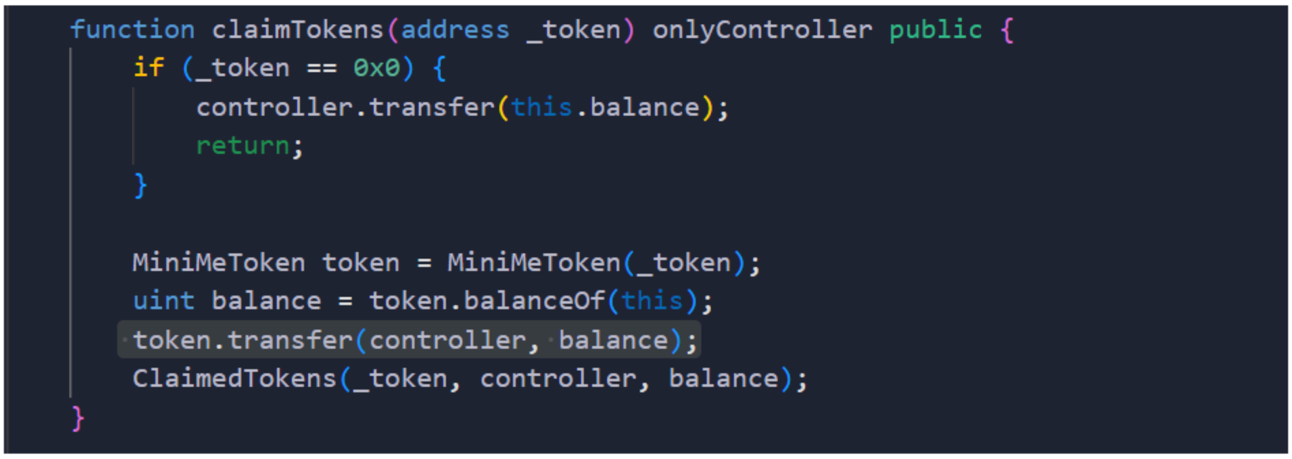
No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

(1) Unchecked-transfer:



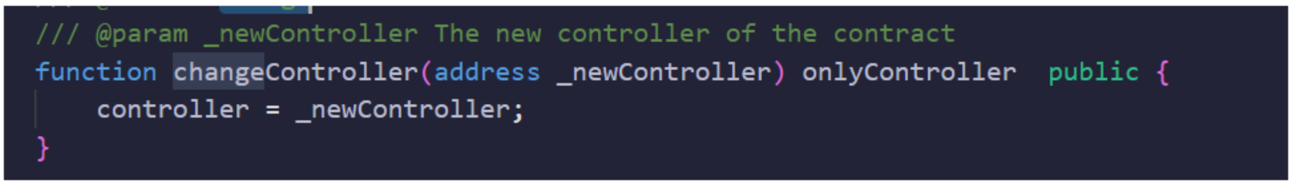
```
function claimTokens(address _token) onlyController public {
    if (_token == 0x0) {
        controller.transfer(this.balance);
        return;
    }

    MiniMeToken token = MiniMeToken(_token);
    uint balance = token.balanceOf(this);
    token.transfer(controller, balance);
    ClaimedTokens(_token, controller, balance);
}
```

The return value of an external transfer call is not checked.

Resolution: Use SafeERC20, or ensure that the transfer return value is checked.

Low

(1) Missing-events-access-control:

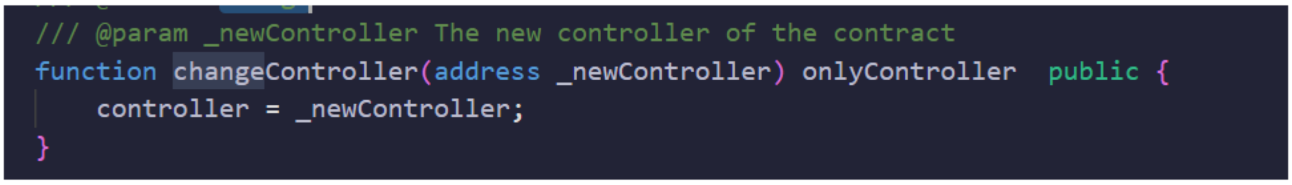


```
/// @param _newController The new controller of the contract
function changeController(address _newController) onlyController  public {
    controller = _newController;
}
```

Detecting missing events for critical access control parameters changeController() has no event, so it is difficult to track off-chain owner changes.

Resolution: We suggest emitting an event for critical parameter changes.

(2) Missing-zero-address-validation:



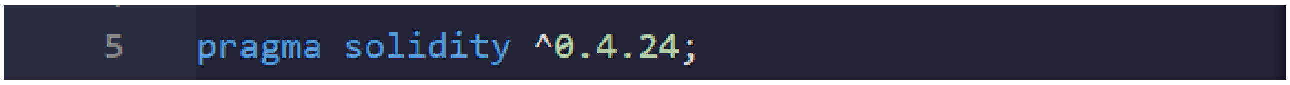
```
/// @param _newController The new controller of the contract
function changeController(address _newController) onlyController  public {
    controller = _newController;
}
```

Detecting missing zero address validation in changeController function. controller can changeController without specifying the _newController, so the owner may lose ownership of the contract.

Resolution: We suggest at first Check that the address is not zero.

Very Low / Informational / Best practices:

(1) Use latest solidity version:

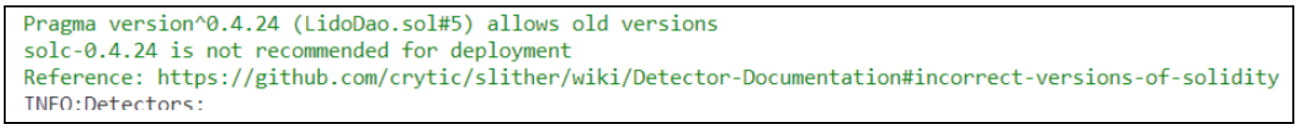


```
pragma solidity ^0.4.24;
```

Use the latest solidity version that protects against any compiler level bugs.

Resolution: Please use versions greater than 0.8.7.

(2) Solc-version:



```
Pragma version^0.4.24 (LidoDao.sol#5) allows old versions
solc-0.4.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
```

Solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

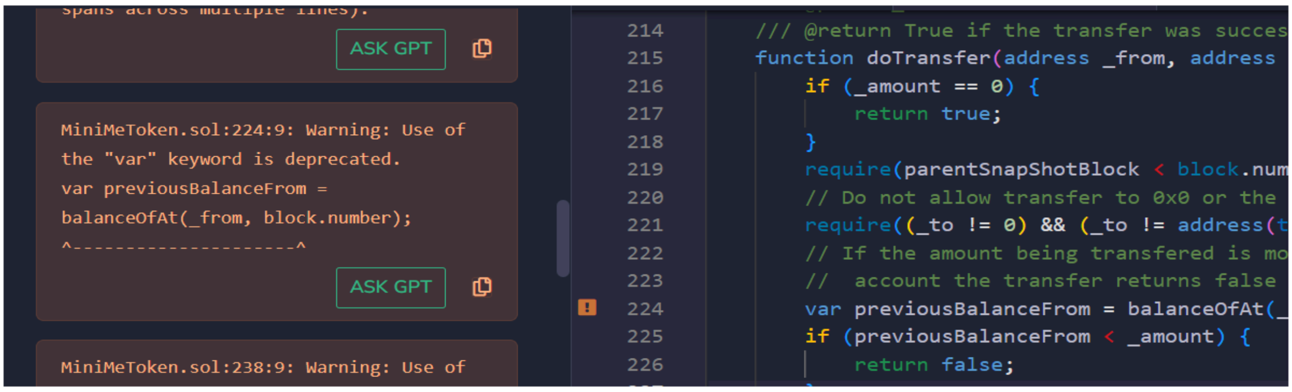
Resolution: Deploy with any of the following Solidity versions:

0.8.18 The recommendations take into account:

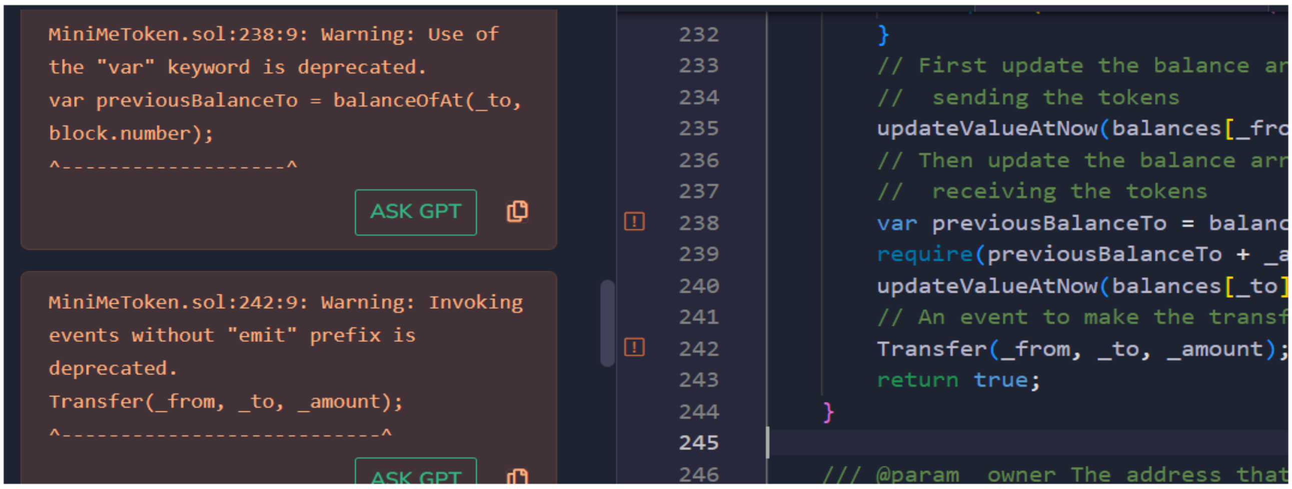
- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

(3) Warning:


```
MiniMeToken.sol:224:9: Warning: Use of
the "var" keyword is deprecated.
var previousBalanceFrom =
balanceOfAt(_from, block.number);
^---------------------^
```

```
MiniMeToken.sol:238:9: Warning: Use of
```
```
/// @return True if the transfer was succes
function doTransfer(address _from, address
    if (_amount == 0) {
        return true;
    }
    require(parentSnapShotBlock < block.num
    // Do not allow transfer to 0x0 or the
    require((_to != 0) && (_to != address(t
    // If the amount being transfered is mo
    //  account the transfer returns false
    var previousBalanceFrom = balanceOfAt(_
    if (previousBalanceFrom < _amount) {
        return false;
```

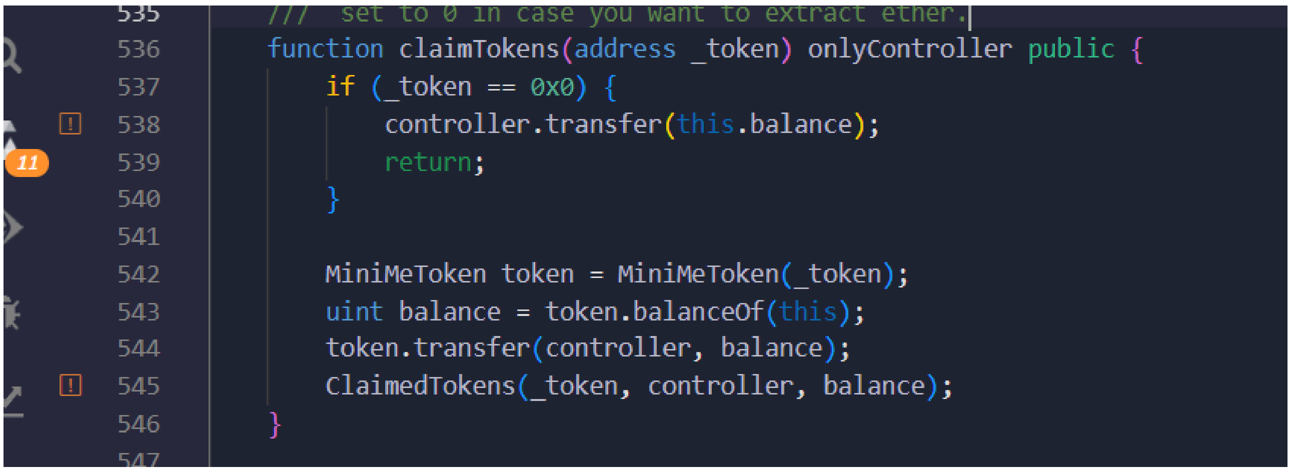

```
MiniMeToken.sol:238:9: Warning: Use of
the "var" keyword is deprecated.
var previousBalanceTo = balanceOfAt(_to,
block.number);
^-------------------^
```
```
MiniMeToken.sol:242:9: Warning: Invoking
events without "emit" prefix is
deprecated.
Transfer(_from, _to, _amount);
^---------------------------^
```

```
            }
            // First update the balance ar
            //  sending the tokens
            updateValueAtNow(balances[_fro
            // Then update the balance arr
            //  receiving the tokens
            var previousBalanceTo = balanc
            require(previousBalanceTo + _a
            updateValueAtNow(balances[_to]
            // An event to make the transf
            Transfer(_from, _to, _amount);
            return true;
    }

    /// @param _owner The address that
```

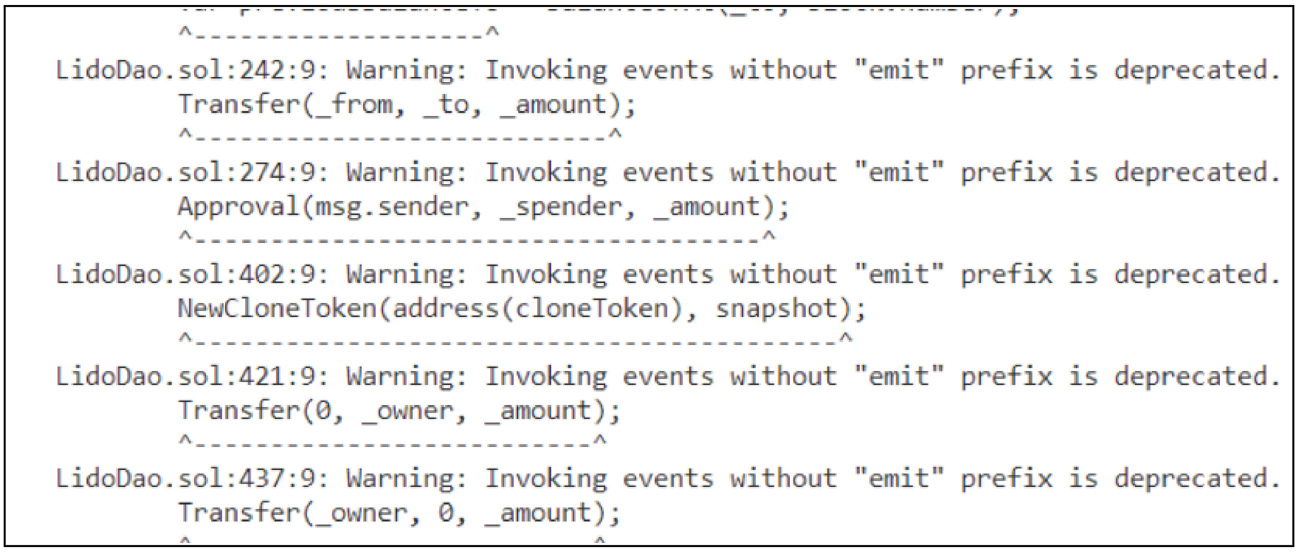
Use of the "var" keyword is deprecated.

Resolution: We suggest please use proper variable names.

https://docs.soliditylang.org/en/v0.4.20/control-structures.html#assignment


```
    ///  set to 0 in case you want to extract ether.
    function claimTokens(address _token) onlyController public {
        if (_token == 0x0) {
            controller.transfer(this.balance);
            return;
        }

        MiniMeToken token = MiniMeToken(_token);
        uint balance = token.balanceOf(this);
        token.transfer(controller, balance);
        ClaimedTokens(_token, controller, balance);
    }
```

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

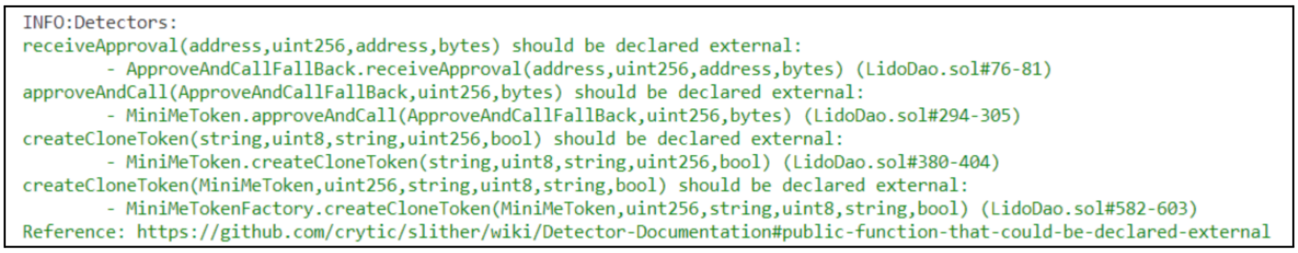


```
        ^-------------------^
LidoDao.sol:242:9: Warning: Invoking events without "emit" prefix is deprecated.
        Transfer(_from, _to, _amount);
        ^---------------------------^
LidoDao.sol:274:9: Warning: Invoking events without "emit" prefix is deprecated.
        Approval(msg.sender, _spender, _amount);
        ^-------------------------------------^
LidoDao.sol:402:9: Warning: Invoking events without "emit" prefix is deprecated.
        NewCloneToken(address(cloneToken), snapshot);
        ^------------------------------------------^
LidoDao.sol:421:9: Warning: Invoking events without "emit" prefix is deprecated.
        Transfer(0, _owner, _amount);
        ^--------------------------^
LidoDao.sol:437:9: Warning: Invoking events without "emit" prefix is deprecated.
        Transfer(_owner, 0, _amount);
        ^                          ^
```

Invoking events without the "emit" prefix is deprecated.

Resolution: We suggest please use proper guidelines of solidity smart contract.

(4) Optimization:



```
INFO:Detectors:
receiveApproval(address,uint256,address,bytes) should be declared external:
        - ApproveAndCallFallBack.receiveApproval(address,uint256,address,bytes) (LidoDao.sol#76-81)
approveAndCall(ApproveAndCallFallBack,uint256,bytes) should be declared external:
        - MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes) (LidoDao.sol#294-305)
createCloneToken(string,uint8,string,uint256,bool) should be declared external:
        - MiniMeToken.createCloneToken(string,uint8,string,uint256,bool) (LidoDao.sol#380-404)
createCloneToken(MiniMeToken,uint256,string,uint8,string,bool) should be declared external:
        - MiniMeTokenFactory.createCloneToken(MiniMeToken,uint256,string,uint8,string,bool) (LidoDao.sol#582-603)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Public functions that are never called by the contract should be declared external, and its immutable parameters should be located in calldata to save gas.

Declare these function external:

- receiveApproval()
- approveAndCall()
- createCloneToken()
- createCloneToken()

Resolution: Use the external attribute for functions never called from the contract, and change the location of immutable parameters to calldata to save gas.

(5) Centralization:

In the contract onlyController as a owner authority on the following function:

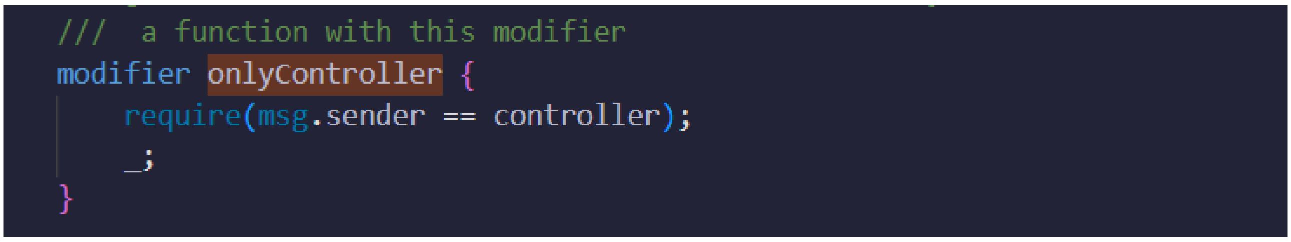
- changeController

- generateTokens
- destroyTokens
- enableTransfers
- claimTokens

Any compromise to these accounts may allow the hacker to manipulate the project through these functions.

Resolution: We suggest carefully managing the owner account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practice.

(6) Error message for require():



```
///  a function with this modifier
modifier onlyController {
    require(msg.sender == controller);
    _;
}
```

Detects missing error message in required statement.

Resolution: We suggest writing a proper message in require(), Otherwise the user can't identify the actual problem why the transaction is not successful.

Centralization

This smart contract has 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:

MiniMeToken.sol

- generateTokens: Generates `_amount` tokens by the controller owner.
- destroyTokens: Burns `_amount` tokens from `_owner` by the controller owner.
- enableTransfers: Enables token holders to transfer their tokens freely if true by the controller owner.
- _transfersEnabled True if transfers are allowed in the clone by the controller owner.
- claimTokens: extract mistakenly sent tokens to this contract by the controller owner.
- changeController: Changes the controller of the contract by the controller owner.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of Etherscan web links. And we have used all possible tests based on given objects as files. We had observed 1 medium, 2 low and 6 informational issues in the smart contracts. And those issues are not critical. So, **it's good to go for the production**.

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

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

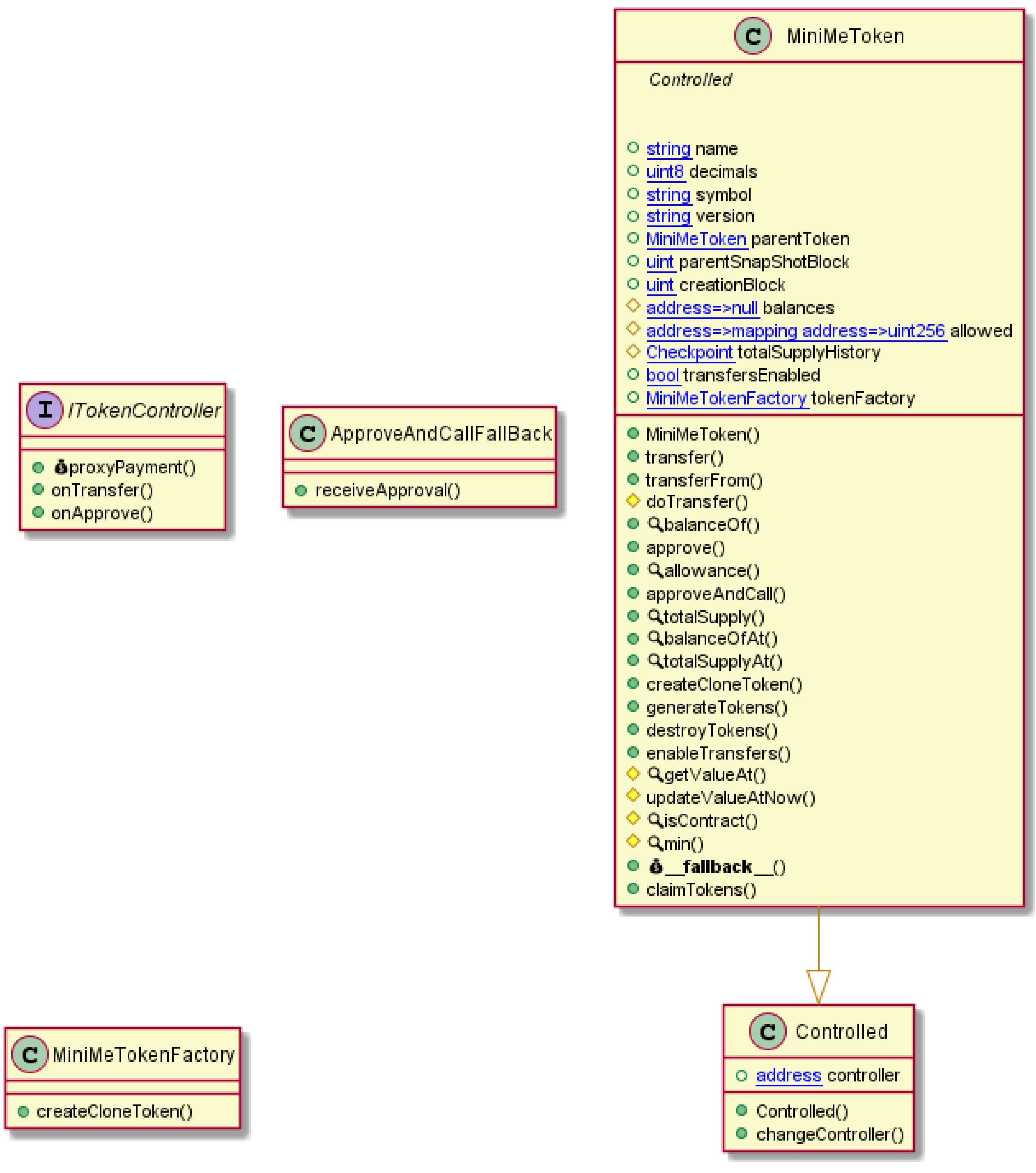
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

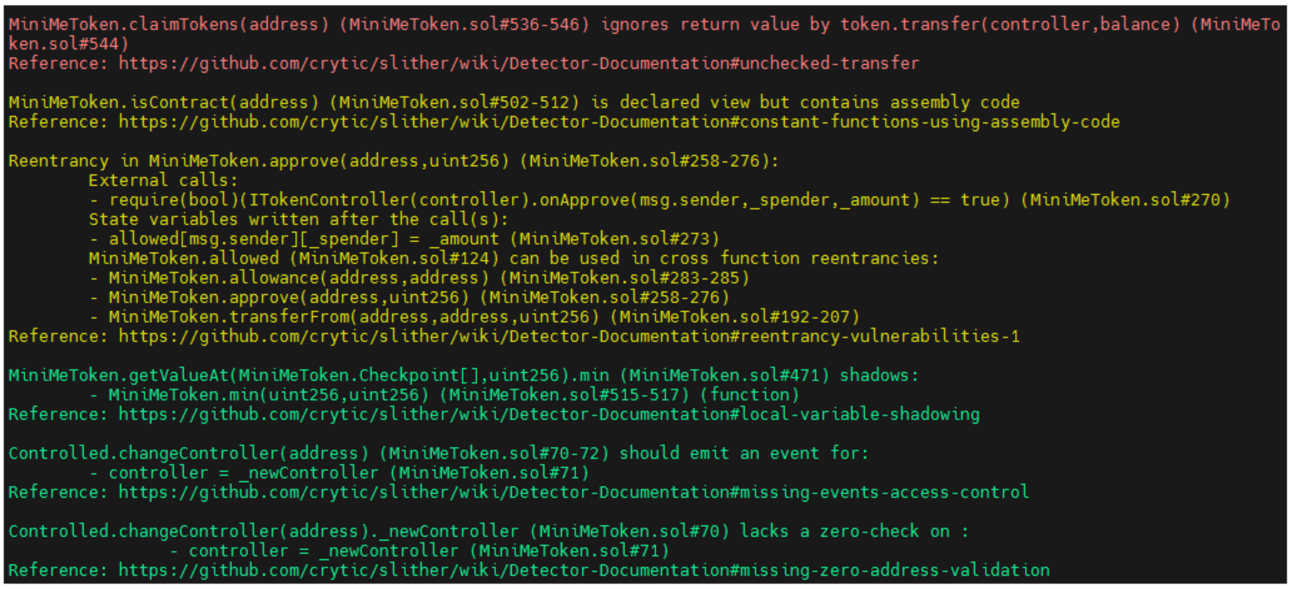


Slither Results Log

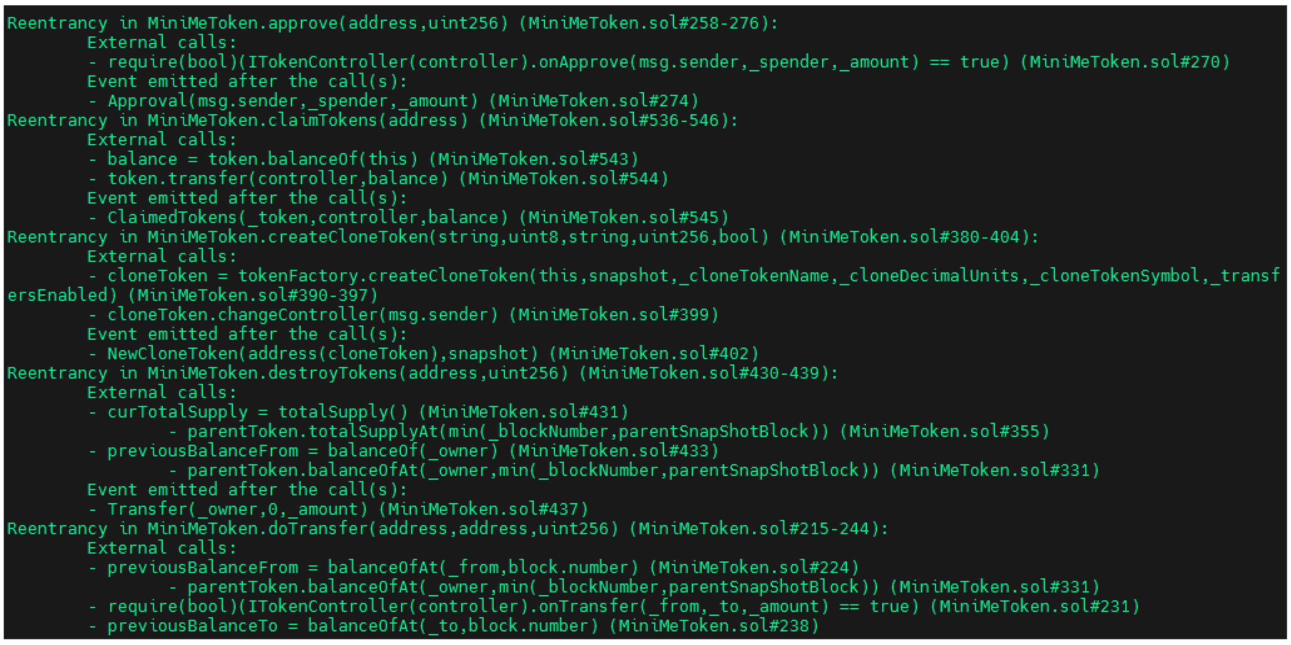
Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

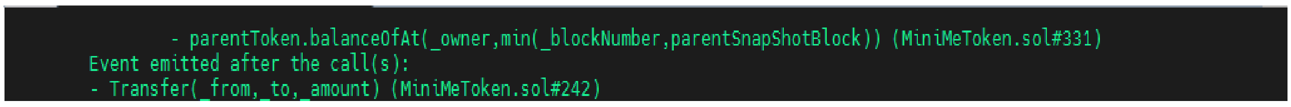
Slither Log >> MiniMeToken.sol



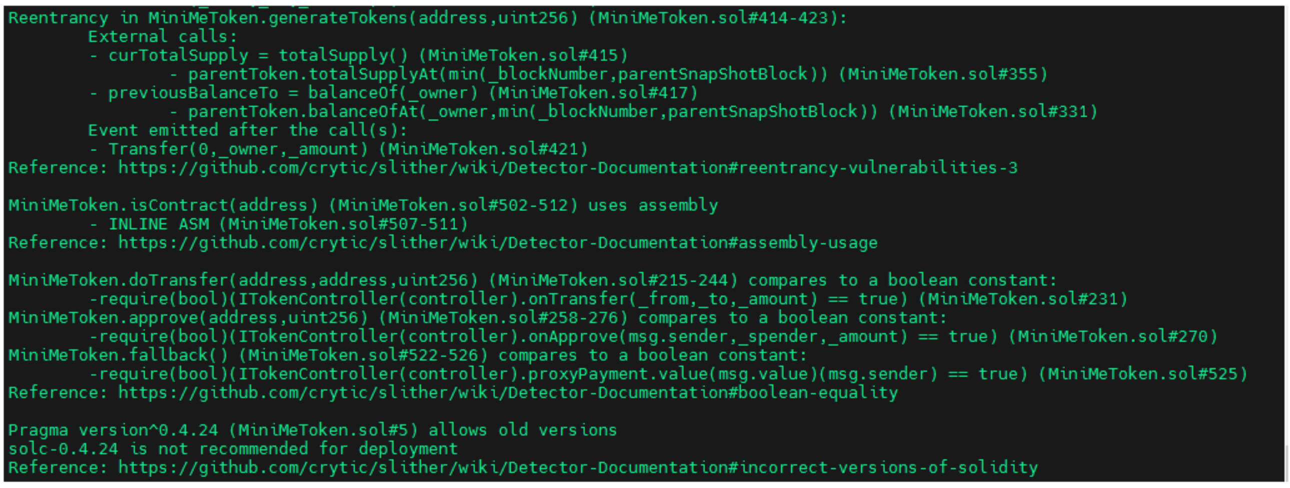
```
MiniMeToken.claimTokens(address) (MiniMeToken.sol#536-546) ignores return value by token.transfer(controller,balance) (MiniMeTo
ken.sol#544)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

MiniMeToken.isContract(address) (MiniMeToken.sol#502-512) is declared view but contains assembly code
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#constant-functions-using-assembly-code

Reentrancy in MiniMeToken.approve(address,uint256) (MiniMeToken.sol#258-276):
        External calls:
        - require(bool)(ITokenController(controller).onApprove(msg.sender,_spender,_amount) == true) (MiniMeToken.sol#270)
        State variables written after the call(s):
        - allowed[msg.sender][_spender] = _amount (MiniMeToken.sol#273)
        MiniMeToken.allowed (MiniMeToken.sol#124) can be used in cross function reentrancies:
        - MiniMeToken.allowance(address,address) (MiniMeToken.sol#283-285)
        - MiniMeToken.approve(address,uint256) (MiniMeToken.sol#258-276)
        - MiniMeToken.transferFrom(address,address,uint256) (MiniMeToken.sol#192-207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

MiniMeToken.getValueAt(MiniMeToken.Checkpoint[],uint256).min (MiniMeToken.sol#471) shadows:
        - MiniMeToken.min(uint256,uint256) (MiniMeToken.sol#515-517) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Controlled.changeController(address) (MiniMeToken.sol#70-72) should emit an event for:
        - controller = _newController (MiniMeToken.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Controlled.changeController(address)._newController (MiniMeToken.sol#70) lacks a zero-check on :
                - controller = _newController (MiniMeToken.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



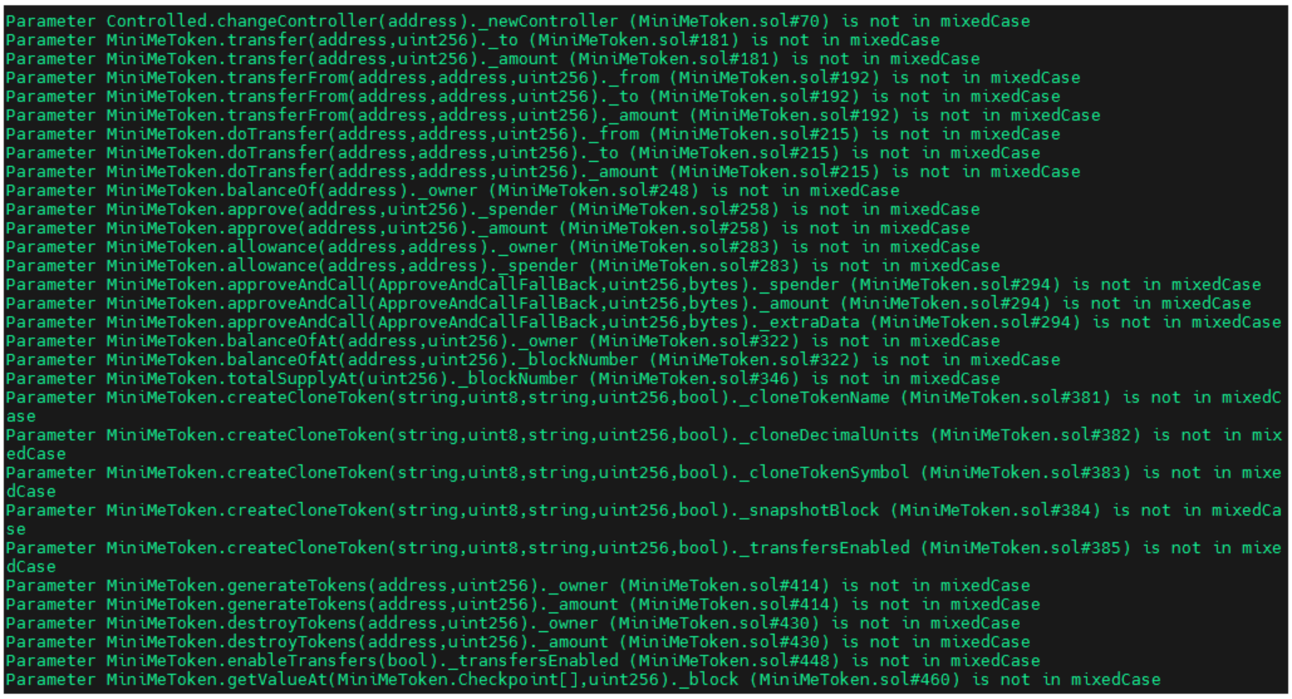
```
Reentrancy in MiniMeToken.approve(address,uint256) (MiniMeToken.sol#258-276):
        External calls:
        - require(bool)(ITokenController(controller).onApprove(msg.sender,_spender,_amount) == true) (MiniMeToken.sol#270)
        Event emitted after the call(s):
        - Approval(msg.sender,_spender,_amount) (MiniMeToken.sol#274)
Reentrancy in MiniMeToken.claimTokens(address) (MiniMeToken.sol#536-546):
        External calls:
        - balance = token.balanceOf(this) (MiniMeToken.sol#543)
        - token.transfer(controller,balance) (MiniMeToken.sol#544)
        Event emitted after the call(s):
        - ClaimedTokens(_token,controller,balance) (MiniMeToken.sol#545)
Reentrancy in MiniMeToken.createCloneToken(string,uint8,string,uint256,bool) (MiniMeToken.sol#380-404):
        External calls:
        - cloneToken = tokenFactory.createCloneToken(this,snapshot,_cloneTokenName,_cloneDecimalUnits,_cloneTokenSymbol,_transf
ersEnabled) (MiniMeToken.sol#390-397)
        - cloneToken.changeController(msg.sender) (MiniMeToken.sol#399)
        Event emitted after the call(s):
        - NewCloneToken(address(cloneToken),snapshot) (MiniMeToken.sol#402)
Reentrancy in MiniMeToken.destroyTokens(address,uint256) (MiniMeToken.sol#430-439):
        External calls:
        - curTotalSupply = totalSupply() (MiniMeToken.sol#431)
                - parentToken.totalSupplyAt(min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#355)
        - previousBalanceFrom = balanceOf(_owner) (MiniMeToken.sol#433)
                - parentToken.balanceOfAt(_owner,min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#331)
        Event emitted after the call(s):
        - Transfer(_owner,0,_amount) (MiniMeToken.sol#437)
Reentrancy in MiniMeToken.doTransfer(address,address,uint256) (MiniMeToken.sol#215-244):
        External calls:
        - previousBalanceFrom = balanceOfAt(_from,block.number) (MiniMeToken.sol#224)
                - parentToken.balanceOfAt(_owner,min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#331)
        - require(bool)(ITokenController(controller).onTransfer(_from,_to,_amount) == true) (MiniMeToken.sol#231)
        - previousBalanceTo = balanceOfAt(_to,block.number) (MiniMeToken.sol#238)
```



```
                - parentToken.balanceOfAt(_owner,min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#331)
        Event emitted after the call(s):
        - Transfer(_from,_to,_amount) (MiniMeToken.sol#242)
```



```
Reentrancy in MiniMeToken.generateTokens(address,uint256) (MiniMeToken.sol#414-423):
        External calls:
        - curTotalSupply = totalSupply() (MiniMeToken.sol#415)
                - parentToken.totalSupplyAt(min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#355)
        - previousBalanceTo = balanceOf(_owner) (MiniMeToken.sol#417)
                - parentToken.balanceOfAt(_owner,min(_blockNumber,parentSnapShotBlock)) (MiniMeToken.sol#331)
        Event emitted after the call(s):
        - Transfer(0,_owner,_amount) (MiniMeToken.sol#421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

MiniMeToken.isContract(address) (MiniMeToken.sol#502-512) uses assembly
        - INLINE ASM (MiniMeToken.sol#507-511)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

MiniMeToken.doTransfer(address,address,uint256) (MiniMeToken.sol#215-244) compares to a boolean constant:
        -require(bool)(ITokenController(controller).onTransfer(_from,_to,_amount) == true) (MiniMeToken.sol#231)
MiniMeToken.approve(address,uint256) (MiniMeToken.sol#258-276) compares to a boolean constant:
        -require(bool)(ITokenController(controller).onApprove(msg.sender,_spender,_amount) == true) (MiniMeToken.sol#270)
MiniMeToken.fallback() (MiniMeToken.sol#522-526) compares to a boolean constant:
        -require(bool)(ITokenController(controller).proxyPayment.value(msg.value)(msg.sender) == true) (MiniMeToken.sol#525)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version^0.4.24 (MiniMeToken.sol#5) allows old versions
solc-0.4.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



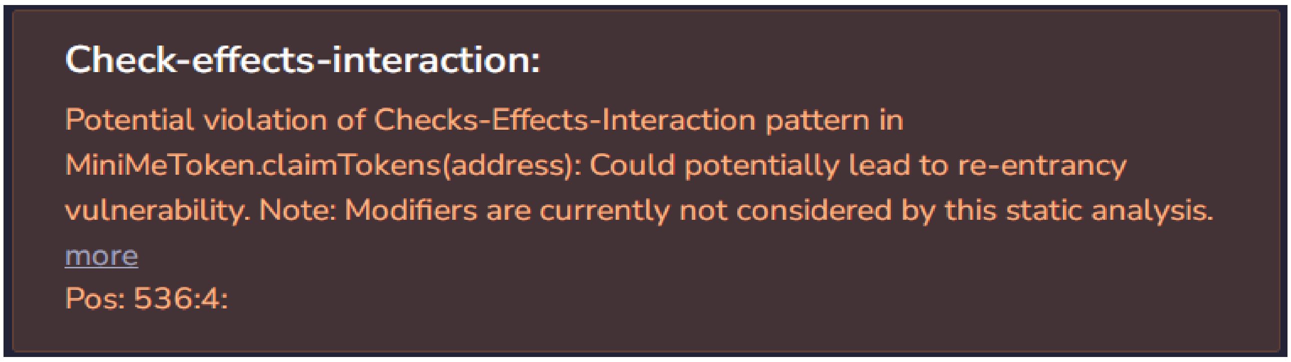
```
Parameter Controlled.changeController(address)._newController (MiniMeToken.sol#70) is not in mixedCase
Parameter MiniMeToken.transfer(address,uint256)._to (MiniMeToken.sol#181) is not in mixedCase
Parameter MiniMeToken.transfer(address,uint256)._amount (MiniMeToken.sol#181) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._from (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._to (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._amount (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._from (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._to (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._amount (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.balanceOf(address)._owner (MiniMeToken.sol#248) is not in mixedCase
Parameter MiniMeToken.approve(address,uint256)._spender (MiniMeToken.sol#258) is not in mixedCase
Parameter MiniMeToken.approve(address,uint256)._amount (MiniMeToken.sol#258) is not in mixedCase
Parameter MiniMeToken.allowance(address,address)._owner (MiniMeToken.sol#283) is not in mixedCase
Parameter MiniMeToken.allowance(address,address)._spender (MiniMeToken.sol#283) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._spender (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._amount (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._extraData (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.balanceOfAt(address,uint256)._owner (MiniMeToken.sol#322) is not in mixedCase
Parameter MiniMeToken.balanceOfAt(address,uint256)._blockNumber (MiniMeToken.sol#322) is not in mixedCase
Parameter MiniMeToken.totalSupplyAt(uint256)._blockNumber (MiniMeToken.sol#346) is not in mixedCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneTokenName (MiniMeToken.sol#381) is not in mixedC
ase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneDecimalUnits (MiniMeToken.sol#382) is not in mix
edCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneTokenSymbol (MiniMeToken.sol#383) is not in mixe
dCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._snapshotBlock (MiniMeToken.sol#384) is not in mixedCa
se
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._transfersEnabled (MiniMeToken.sol#385) is not in mixe
dCase
Parameter MiniMeToken.generateTokens(address,uint256)._owner (MiniMeToken.sol#414) is not in mixedCase
Parameter MiniMeToken.generateTokens(address,uint256)._amount (MiniMeToken.sol#414) is not in mixedCase
Parameter MiniMeToken.destroyTokens(address,uint256)._owner (MiniMeToken.sol#430) is not in mixedCase
Parameter MiniMeToken.destroyTokens(address,uint256)._amount (MiniMeToken.sol#430) is not in mixedCase
Parameter MiniMeToken.enableTransfers(bool)._transfersEnabled (MiniMeToken.sol#448) is not in mixedCase
Parameter MiniMeToken.getValueAt(MiniMeToken.Checkpoint[],uint256)._block (MiniMeToken.sol#460) is not in mixedCase
```

```
Parameter Controlled.changeController(address)._newController (MiniMeToken.sol#70) is not in mixedCase
Parameter MiniMeToken.transfer(address,uint256)._to (MiniMeToken.sol#181) is not in mixedCase
Parameter MiniMeToken.transfer(address,uint256)._amount (MiniMeToken.sol#181) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._from (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._to (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.transferFrom(address,address,uint256)._amount (MiniMeToken.sol#192) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._from (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._to (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.doTransfer(address,address,uint256)._amount (MiniMeToken.sol#215) is not in mixedCase
Parameter MiniMeToken.balanceOf(address)._owner (MiniMeToken.sol#248) is not in mixedCase
Parameter MiniMeToken.approve(address,uint256)._spender (MiniMeToken.sol#258) is not in mixedCase
Parameter MiniMeToken.approve(address,uint256)._amount (MiniMeToken.sol#258) is not in mixedCase
Parameter MiniMeToken.allowance(address,address)._owner (MiniMeToken.sol#283) is not in mixedCase
Parameter MiniMeToken.allowance(address,address)._spender (MiniMeToken.sol#283) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._spender (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._amount (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.approveAndCall(ApproveAndCallFallBack,uint256,bytes)._extraData (MiniMeToken.sol#294) is not in mixedCase
Parameter MiniMeToken.balanceOfAt(address,uint256)._owner (MiniMeToken.sol#322) is not in mixedCase
Parameter MiniMeToken.balanceOfAt(address,uint256)._blockNumber (MiniMeToken.sol#322) is not in mixedCase
Parameter MiniMeToken.totalSupplyAt(uint256)._blockNumber (MiniMeToken.sol#346) is not in mixedCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneTokenName (MiniMeToken.sol#381) is not in mixedC
ase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneDecimalUnits (MiniMeToken.sol#382) is not in mix
edCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._cloneTokenSymbol (MiniMeToken.sol#383) is not in mixe
dCase
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._snapshotBlock (MiniMeToken.sol#384) is not in mixedCa
se
Parameter MiniMeToken.createCloneToken(string,uint8,string,uint256,bool)._transfersEnabled (MiniMeToken.sol#385) is not in mixe
dCase
Parameter MiniMeToken.generateTokens(address,uint256)._owner (MiniMeToken.sol#414) is not in mixedCase
Parameter MiniMeToken.generateTokens(address,uint256)._amount (MiniMeToken.sol#414) is not in mixedCase
Parameter MiniMeToken.destroyTokens(address,uint256)._owner (MiniMeToken.sol#430) is not in mixedCase
Parameter MiniMeToken.destroyTokens(address,uint256)._amount (MiniMeToken.sol#430) is not in mixedCase
Parameter MiniMeToken.enableTransfers(bool)._transfersEnabled (MiniMeToken.sol#448) is not in mixedCase
Parameter MiniMeToken.getValueAt(MiniMeToken.Checkpoint[],uint256)._block (MiniMeToken.sol#460) is not in mixedCase
```

Solidity Static Analysis

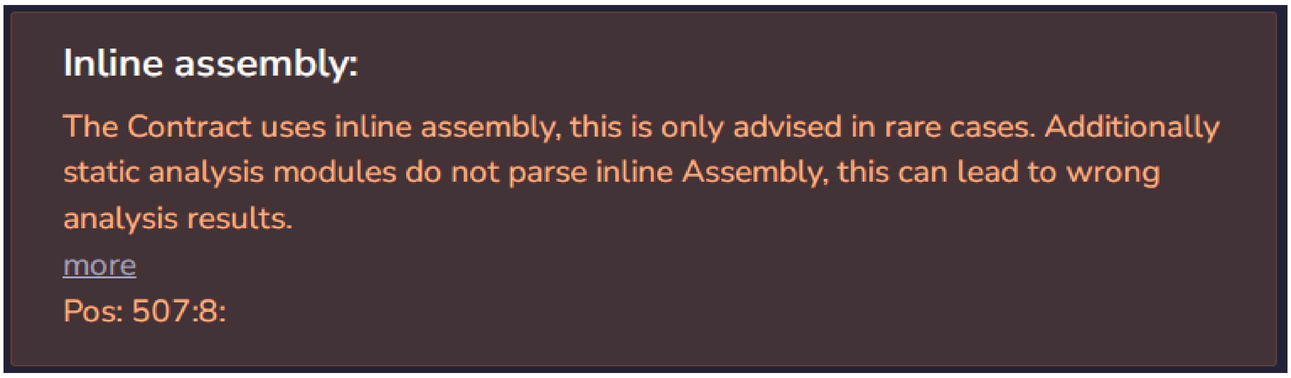
Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

MiniMeToken.sol



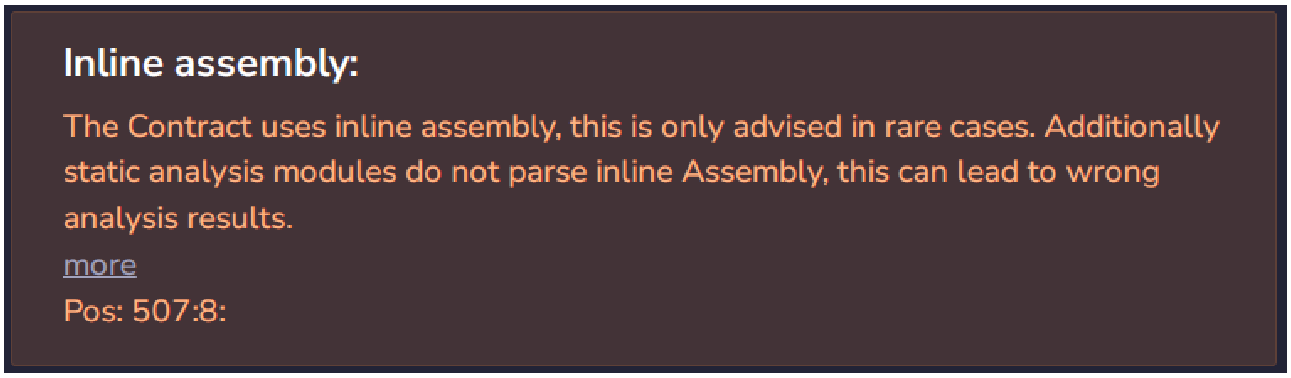
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MiniMeToken.claimTokens(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 536:4:



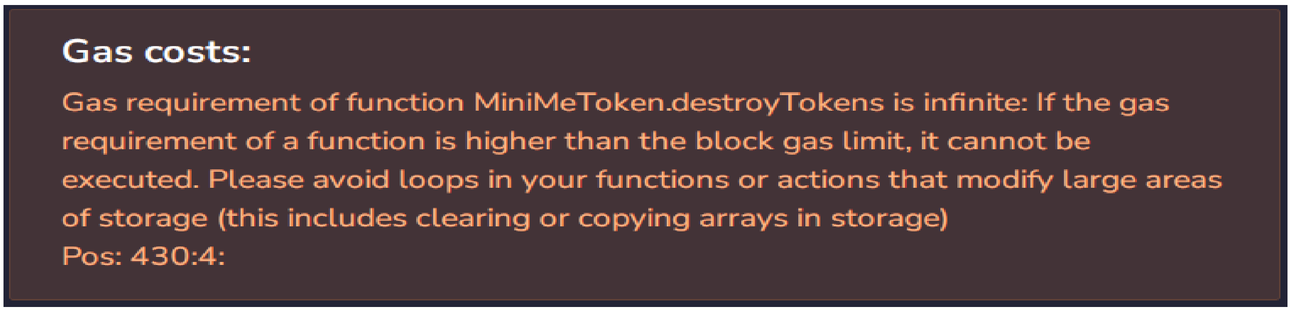
Inline assembly:

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: 507:8:



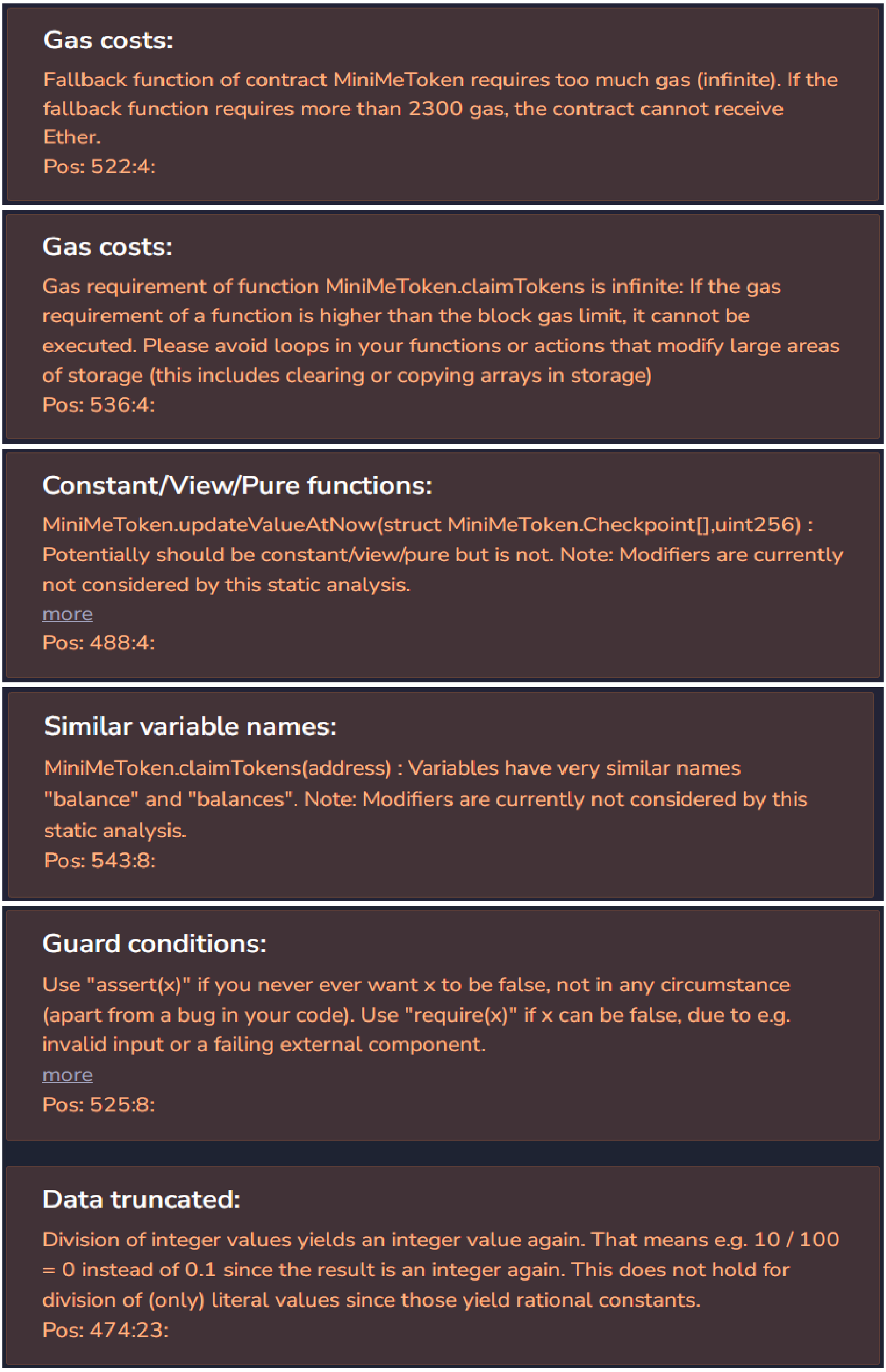
Inline assembly:

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: 507:8:



Gas costs:

Gas requirement of function MiniMeToken.destroyTokens 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: 430:4:



Gas costs:

Fallback function of contract MiniMeToken requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 522:4:

Gas costs:

Gas requirement of function MiniMeToken.claimTokens 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: 536:4:

Constant/View/Pure functions:

MiniMeToken.updateValueAtNow(struct MiniMeToken.Checkpoint[],uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 488:4:

Similar variable names:

MiniMeToken.claimTokens(address) : Variables have very similar names "balance" and "balances". Note: Modifiers are currently not considered by this static analysis.
Pos: 543:8:

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: 525:8:

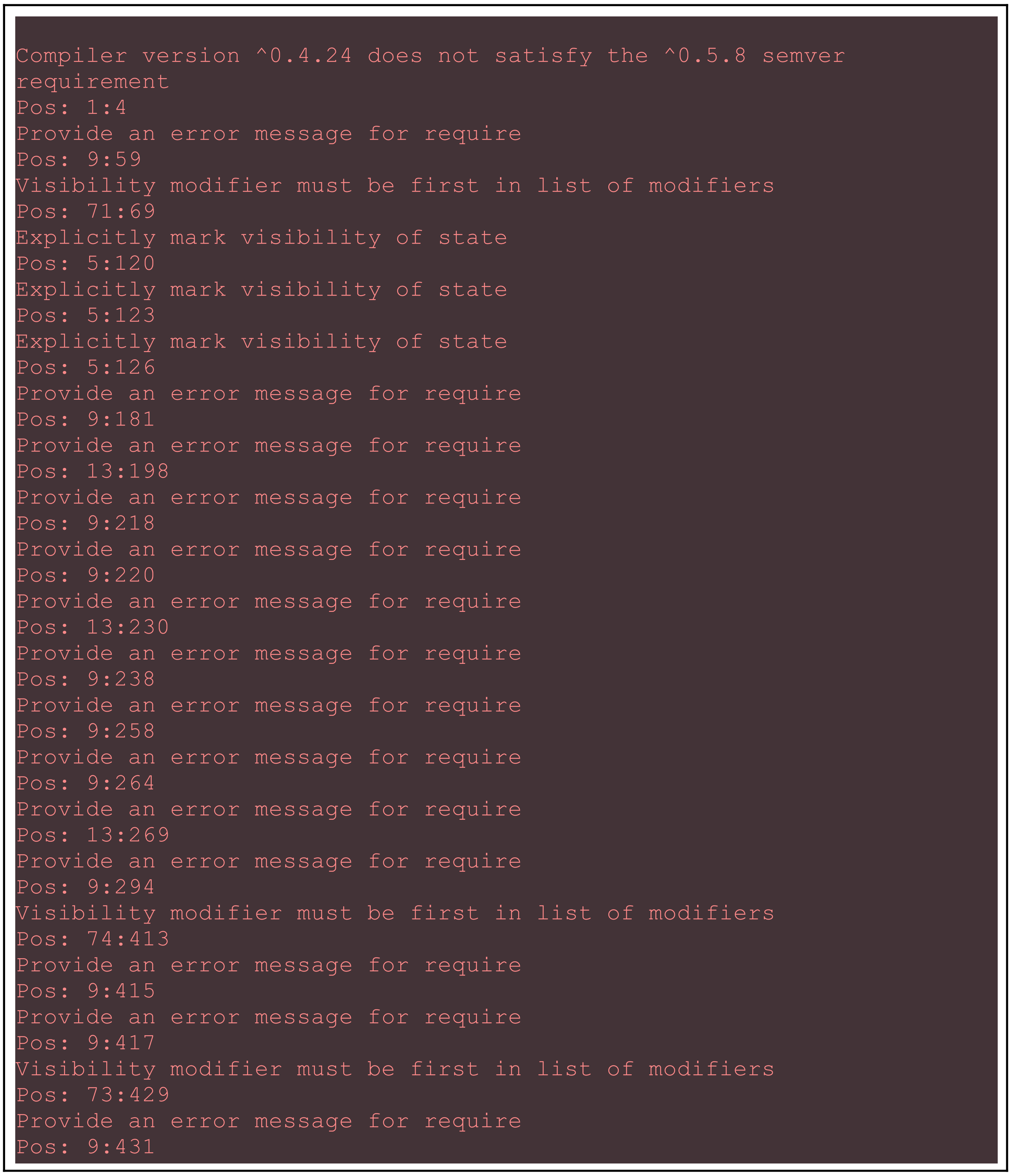
Data truncated:

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: 474:23:

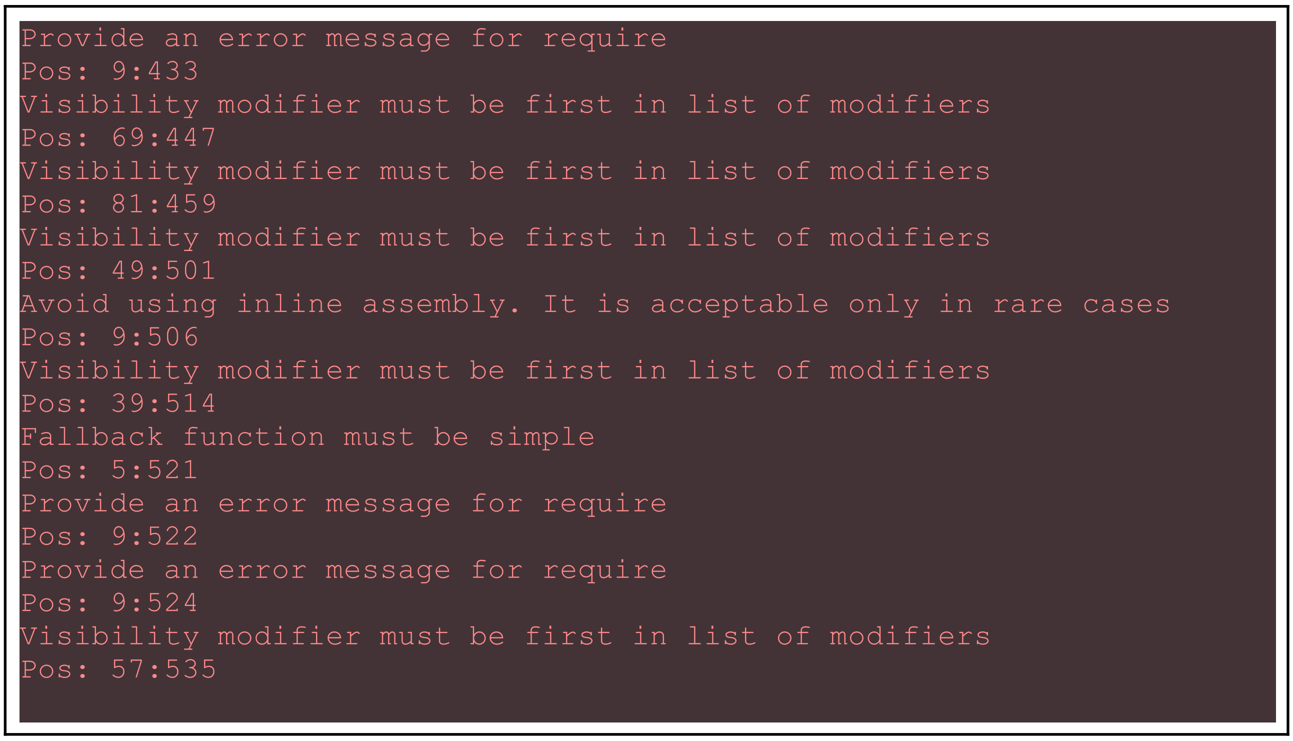
Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

MiniMeToken.sol



```
Compiler version ^0.4.24 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:4
Provide an error message for require
Pos: 9:59
Visibility modifier must be first in list of modifiers
Pos: 71:69
Explicitly mark visibility of state
Pos: 5:120
Explicitly mark visibility of state
Pos: 5:123
Explicitly mark visibility of state
Pos: 5:126
Provide an error message for require
Pos: 9:181
Provide an error message for require
Pos: 13:198
Provide an error message for require
Pos: 9:218
Provide an error message for require
Pos: 9:220
Provide an error message for require
Pos: 13:230
Provide an error message for require
Pos: 9:238
Provide an error message for require
Pos: 9:258
Provide an error message for require
Pos: 9:264
Provide an error message for require
Pos: 13:269
Provide an error message for require
Pos: 9:294
Visibility modifier must be first in list of modifiers
Pos: 74:413
Provide an error message for require
Pos: 9:415
Provide an error message for require
Pos: 9:417
Visibility modifier must be first in list of modifiers
Pos: 73:429
Provide an error message for require
Pos: 9:431
```


```
Provide an error message for require
Pos: 9:433
Visibility modifier must be first in list of modifiers
Pos: 69:447
Visibility modifier must be first in list of modifiers
Pos: 81:459
Visibility modifier must be first in list of modifiers
Pos: 49:501
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:506
Visibility modifier must be first in list of modifiers
Pos: 39:514
Fallback function must be simple
Pos: 5:521
Provide an error message for require
Pos: 9:522
Provide an error message for require
Pos: 9:524
Visibility modifier must be first in list of modifiers
Pos: 57:535
```

Software analysis result:

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