Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Maverick Token
Website:	mav.xyz
Platform:	Ethereum
Language:	Solidity
Date:	May 8th, 2024

Table of contents

`

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.

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the Maverick Token smart contract from mav.xyz was audited extensively. 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 May 8th, 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 MaverickToken contract leverages the LayerZero OFT framework to create an omni chain token named "Maverick Token" with the symbol "MAV".
- The contract initializes with an optional initial mint to a specified address. Key functionalities include interface support, token minting, and burning, with omni chain capabilities provided by Layer Zeros OFT framework.

Audit scope

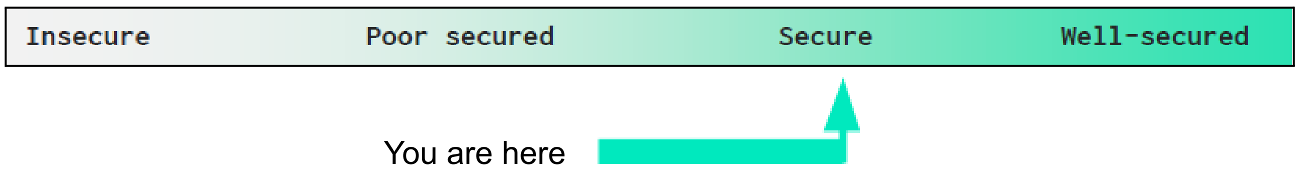
Name	**Code Review and Security Analysis Report for Maverick Token Smart Contract**
Platform	**Ethereum**
Language	**Solidity**
File	MaverickToken.sol
Smart Contract Code	0x7448c7456a97769F6cD04F1E83A4a23cCdC46aBD
Audit Date	May 8th, 2024

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Maverick Token • Symbol: MAV • Decimals: 18	**YES, This is valid.**
Ownership control: • The owner can pause/unpause the contract state. • Burn amount. • Updates account to a frozen state. • Mint a new token. • The current owner can transfer the ownership. • The new owner accepts the ownership transfer.	**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. The 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, 0 medium, 0 low, and 2 very low-level issues.

Investor 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	The solidity version is not specified	Passed
	The solidity version is too old	Passed
	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	Passed
	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?	No
● Max Tax?	No
● Is it Anti-whale?	No
● Is Anti-bot?	Not Detected
● Is it a Blacklist?	Not Detected
● Blacklist Check	No
● Can Mint?	No
● Is it a Proxy?	No
● Can Take Ownership?	Yes
● Hidden Owner?	Not Detected
● Self Destruction?	Not Detected
● Auditor Confidence	High

Overall Audit Result: PASSED

Code Quality

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

The libraries in Maverick 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 Maverick Token.

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 Maverick 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	supportsInterface	read	Passed	No Issue
3	token	read	Passed	No Issue
4	circulatingSupply	read	Passed	No Issue
5	_debitFrom	internal	Passed	No Issue
6	_creditTo	internal	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	estimateSendFee	read	Passed	No Issue
9	sendFrom	write	Passed	No Issue
10	setUseCustomAdapterParams	write	Centralized Ownership and Privileges Management	Refer Audit Findings
11	_nonblockingLzReceive	internal	Passed	No Issue
12	_send	internal	Passed	No Issue
13	_sendAck	internal	Passed	No Issue
14	_checkAdapterParams	internal	Passed	No Issue
15	_debitFrom	internal	Passed	No Issue
16	_creditTo	internal	Passed	No Issue
17	_blockingLzReceive	internal	Passed	No Issue
18	_storeFailedMessage	internal	Passed	No Issue
19	nonblockingLzReceive	write	Passed	No Issue
20	_nonblockingLzReceive	internal	Passed	No Issue
21	retryMessage	write	Passed	No Issue
22	lzReceive	write	Passed	No Issue
23	_blockingLzReceive	internal	Passed	No Issue
24	_lzSend	internal	Passed	No Issue
25	_checkGasLimit	internal	Passed	No Issue
26	_getGasLimit	internal	Passed	No Issue
27	_checkPayloadSize	internal	Passed	No Issue
28	getConfig	external	Passed	No Issue
29	setConfig	external	access only Owner	No Issue
30	setSendVersion	external	Centralized Ownership and Privileges Management	Refer Audit Findings
31	setReceiveVersion	external	Centralized Ownership and Privileges Management	Refer Audit Findings
32	forceResumeReceive	external	Centralized Ownership and Privileges Management	Refer Audit Findings
33	setTrustedRemote	external	access only Owner	No Issue
34	setTrustedRemoteAddress	external	access only Owner	No Issue
35	getTrustedRemoteAddress	external	Passed	No Issue

36	setPrecrime	external	Centralized Ownership and Privileges Management	Refer Audit Findings
37	setMinDstGas	external	Centralized Ownership and Privileges Management	Refer Audit Findings
38	setPayloadSizeLimit	external	Centralized Ownership and Privileges Management	Refer Audit Findings
39	isTrustedRemote	external	Passed	No Issue
40	name	read	Passed	No Issue
41	symbol	read	Passed	No Issue
42	decimals	read	Passed	No Issue
43	totalSupply	read	Passed	No Issue
44	balanceOf	read	Passed	No Issue
45	transfer	write	Passed	No Issue
46	allowance	read	Passed	No Issue
47	approve	write	Passed	No Issue
48	transferFrom	write	Passed	No Issue
49	increaseAllowance	write	Passed	No Issue
50	decreaseAllowance	write	Passed	No Issue
51	_transfer	internal	Passed	No Issue
52	_mint	internal	Passed	No Issue
53	_burn	internal	Passed	No Issue
54	_approve	internal	Passed	No Issue
55	_spendAllowance	internal	Passed	No Issue
56	_beforeTokenTransfer	internal	Passed	No Issue
57	_afterTokenTransfer	internal	Passed	No Issue
58	onlyOwner	modifier	Passed	No Issue
59	owner	read	Passed	No Issue
60	_checkOwner	internal	Passed	No Issue
61	renounceOwnership	write	access only Owner	No Issue
62	transferOwnership	write	access only Owner	No Issue
63	_transferOwnership	internal	Passed	No Issue
64	supportsInterface	read	Passed	No Issue
65	_msgSender	internal	Passed	No Issue
66	_msgData	internal	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.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

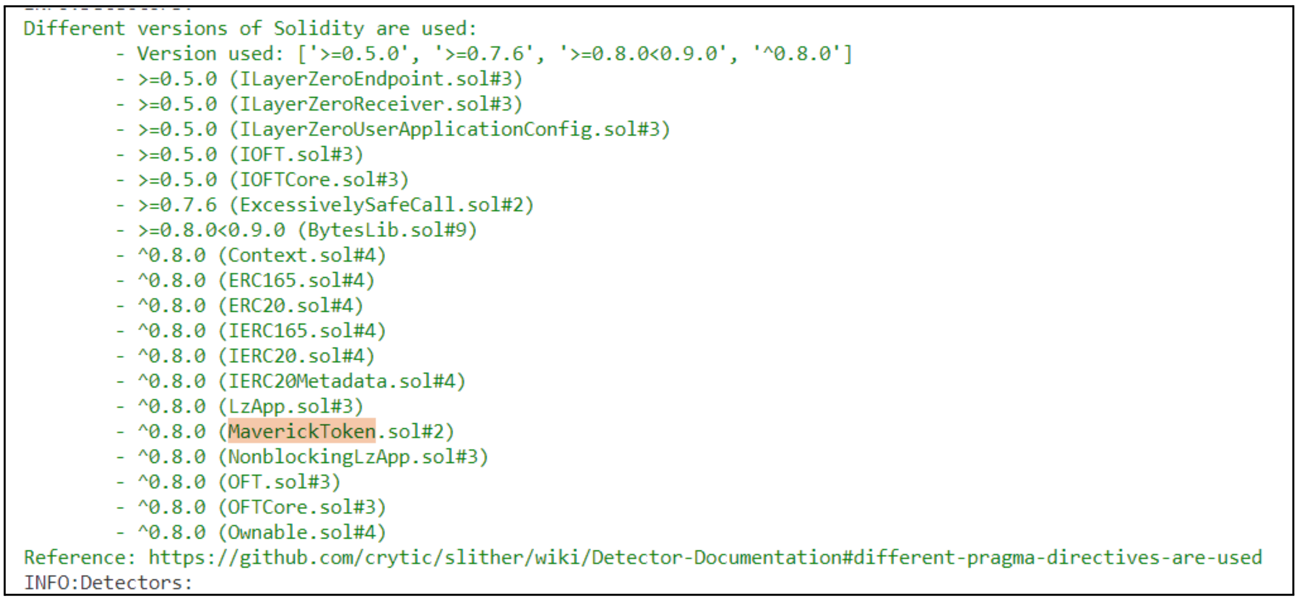
No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best Practices:

(1) Consistent Pragma Solidity Version Usage: **MaverickToken.sol**



```
Different versions of Solidity are used:
        - Version used: ['>=0.5.0', '>=0.7.6', '>=0.8.0<0.9.0', '^0.8.0']
        - >=0.5.0 (ILayerZeroEndpoint.sol#3)
        - >=0.5.0 (ILayerZeroReceiver.sol#3)
        - >=0.5.0 (ILayerZeroUserApplicationConfig.sol#3)
        - >=0.5.0 (IOFT.sol#3)
        - >=0.5.0 (IOFTCore.sol#3)
        - >=0.7.6 (ExcessivelySafeCall.sol#2)
        - >=0.8.0<0.9.0 (BytesLib.sol#9)
        - ^0.8.0 (Context.sol#4)
        - ^0.8.0 (ERC165.sol#4)
        - ^0.8.0 (ERC20.sol#4)
        - ^0.8.0 (IERC165.sol#4)
        - ^0.8.0 (IERC20.sol#4)
        - ^0.8.0 (IERC20Metadata.sol#4)
        - ^0.8.0 (LzApp.sol#3)
        - ^0.8.0 (MaverickToken.sol#2)
        - ^0.8.0 (NonblockingLzApp.sol#3)
        - ^0.8.0 (OFT.sol#3)
        - ^0.8.0 (OFTCore.sol#3)
        - ^0.8.0 (Ownable.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
```

Detected different Solidity versions are used.

Resolution: Use one Solidity version.

(2) Centralized Ownership and Privileges Management:

In the contract, only the owner has owner authority on the following functions:

LzApp.sol

- setSendVersion
- setReceiveVersion
- forceResumeReceive
- setTrustedRemote
- setPrecrime
- setMinDstGas
- setPayloadSizeLimit

OFTCore.sol

- setUseCustomAdapterParams

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

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet's private key would be compromised, then it would create trouble. The following are Admin functions:

OFTCore.sol

- setUseCustomAdapterParams: The owner can update the custom adapter parameter value.
-

LzApp.sol

- lzReceive: The owner can update src chainId and src address.
- setConfig: The owner can update the generic config for the LayerZero user Application.
- setSendVersion: The owner can update the seder version.
- setReceiveVersion: The owner can update the receiver version.
- forceResumeReceive: The owner can forcefully resume the receiver.
- setTrustedRemote: The owner can set the trusted path for the cross-chain communication.
- setTrustedRemoteAddress: The owner can update trusted remote addresses.
- setPrecrime: The owner can update the precrime address.
- setMinDstGas: The owner can update the minimum gas value.
- setPayloadSizeLimit: The owner can update the payload size limit.

Ownable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: The current owner can transfer ownership of the contract to a new account.

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 observed 2 Informational issues in the smart contracts. but those 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 the 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.

The 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 the 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 website 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, and 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).

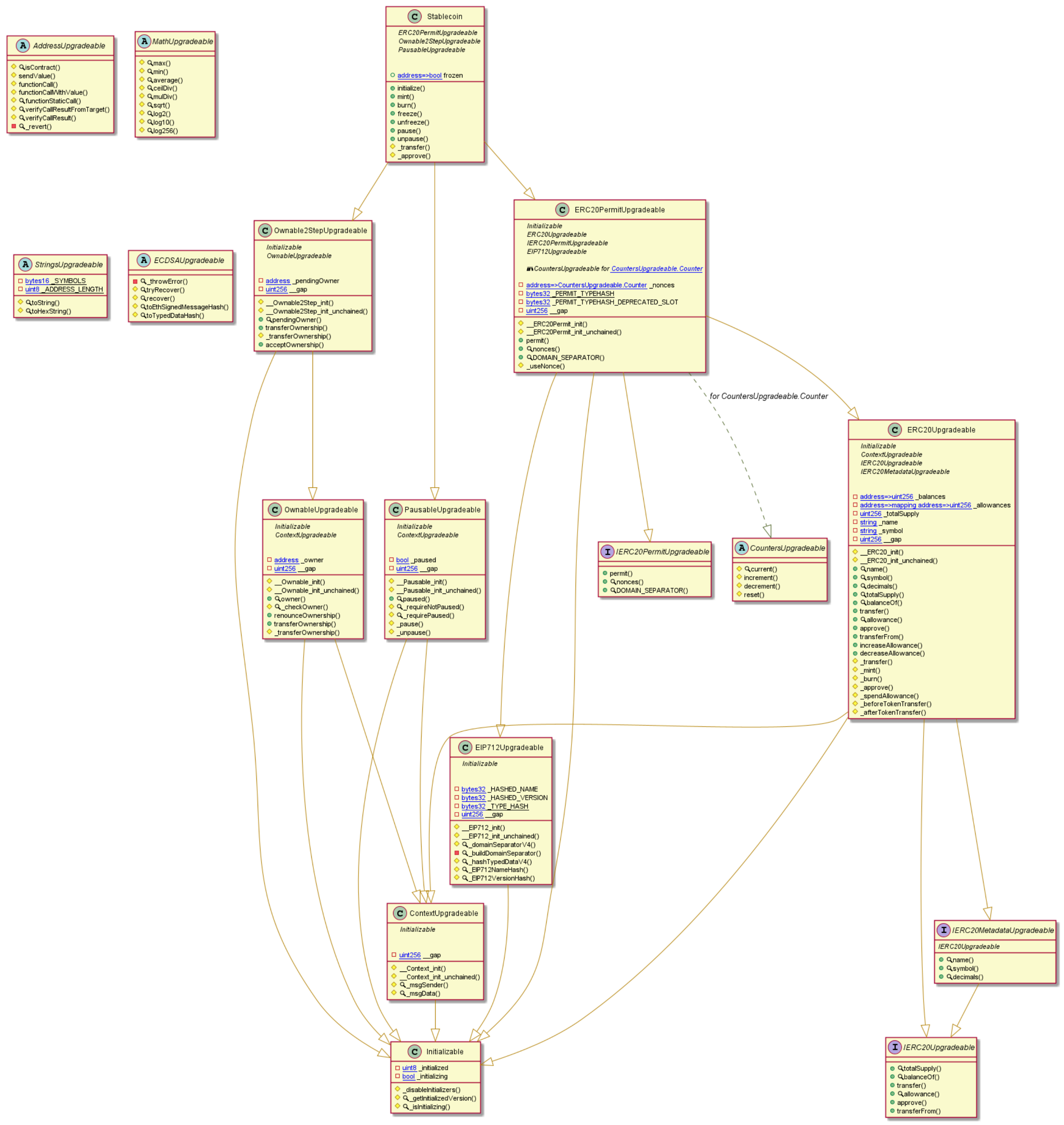
Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status or any other statements of the contract. While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

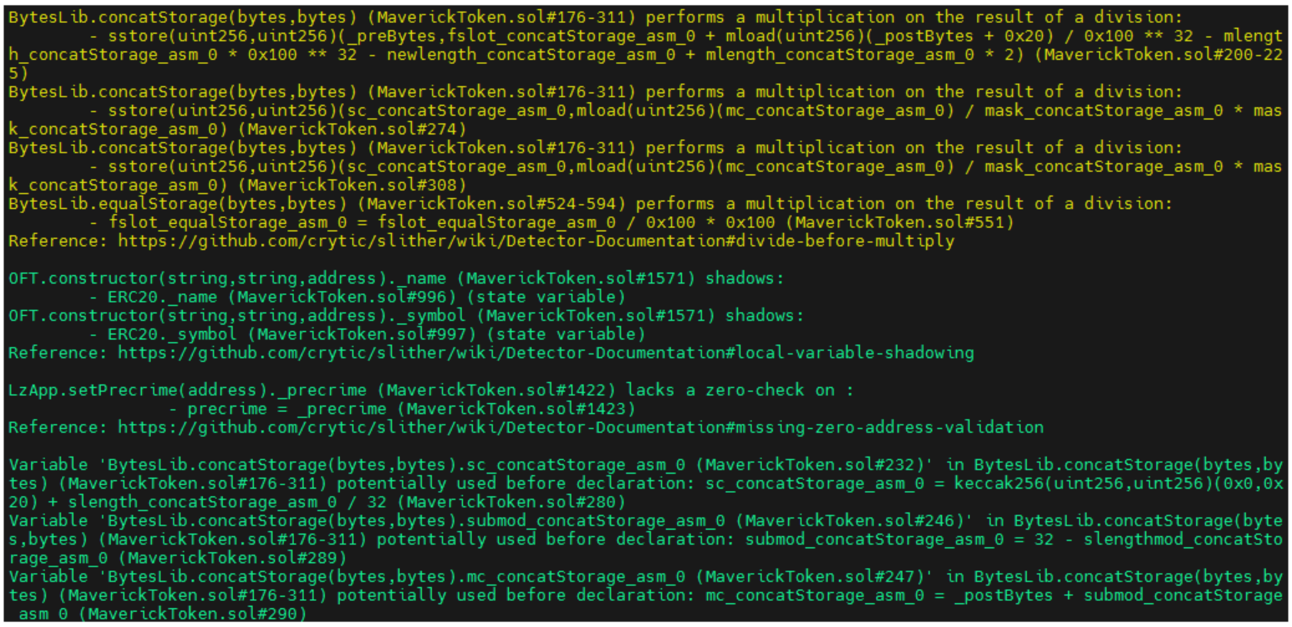


Slither Results Log

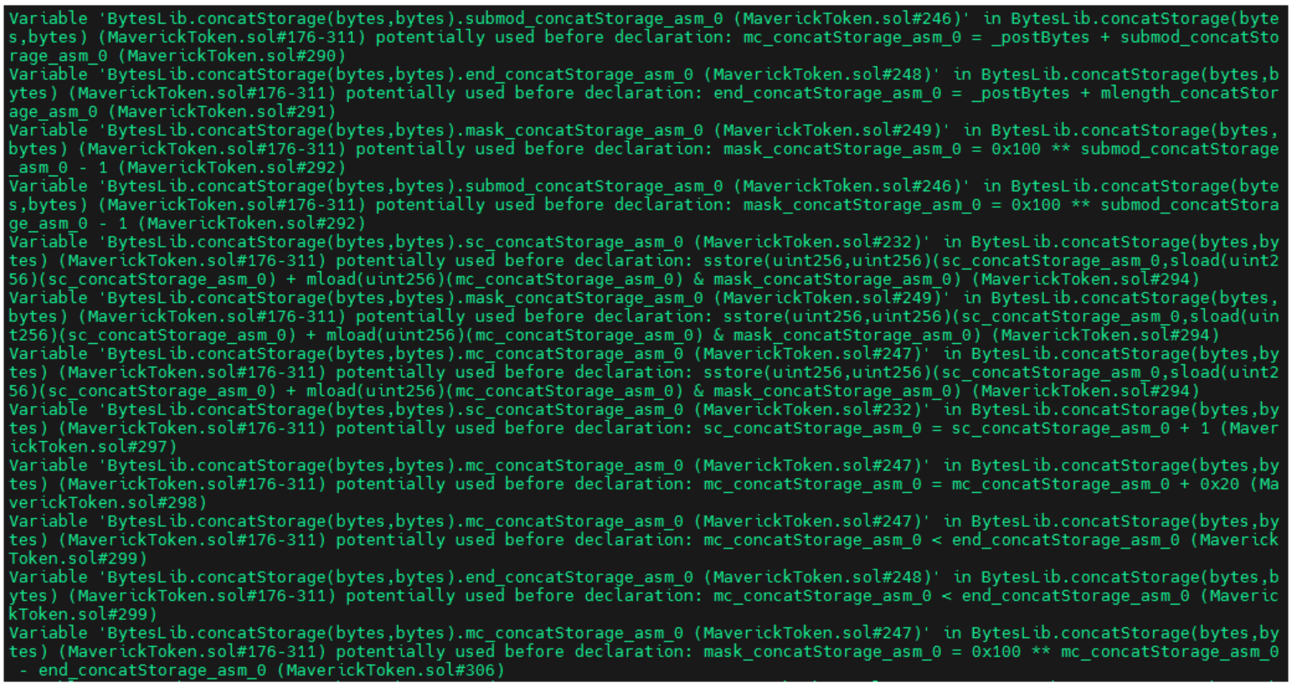
Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report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

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

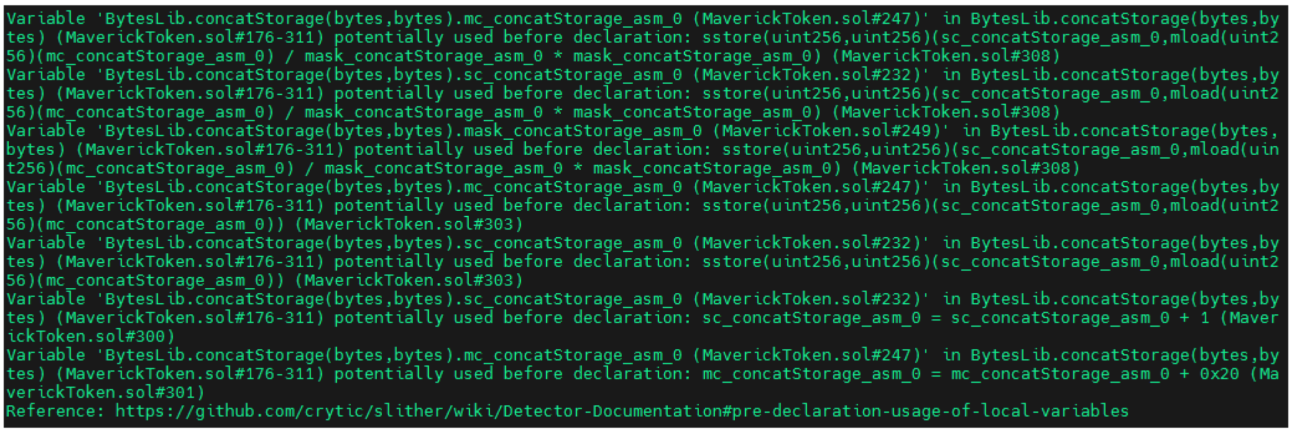
Slither Log >> MaverickToken.sol



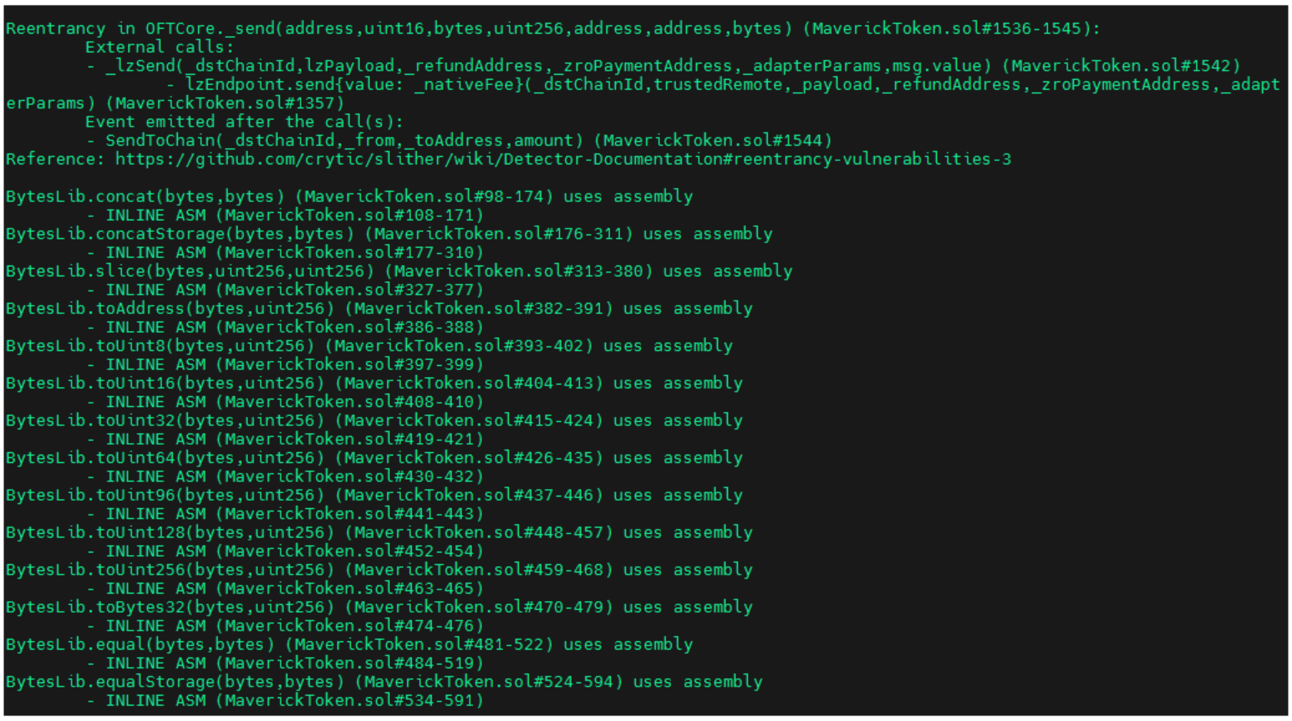
```
BytesLib.concatStorage(bytes,bytes) (MaverickToken.sol#176-311) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(_preBytes,fslot_concatStorage_asm_0 + mload(uint256)(_postBytes + 0x20) / 0x100 ** 32 - mlengt
h_concatStorage_asm_0 * 0x100 ** 32 - newlength_concatStorage_asm_0 + mlength_concatStorage_asm_0 * 2) (MaverickToken.sol#200-22
5)
BytesLib.concatStorage(bytes,bytes) (MaverickToken.sol#176-311) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mas
k_concatStorage_asm_0) (MaverickToken.sol#274)
BytesLib.concatStorage(bytes,bytes) (MaverickToken.sol#176-311) performs a multiplication on the result of a division:
        - sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mas
k_concatStorage_asm_0) (MaverickToken.sol#308)
BytesLib.equalStorage(bytes,bytes) (MaverickToken.sol#524-594) performs a multiplication on the result of a division:
        - fslot_equalStorage_asm_0 = fslot_equalStorage_asm_0 / 0x100 * 0x100 (MaverickToken.sol#551)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

OFT.constructor(string,string,address)._name (MaverickToken.sol#1571) shadows:
        - ERC20._name (MaverickToken.sol#996) (state variable)
OFT.constructor(string,string,address)._symbol (MaverickToken.sol#1571) shadows:
        - ERC20._symbol (MaverickToken.sol#997) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

LzApp.setPrecrime(address)._precrime (MaverickToken.sol#1422) lacks a zero-check on :
                - precrime = _precrime (MaverickToken.sol#1423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sc_concatStorage_asm_0 = keccak256(uint256,uint256)(0x0,0x
20) + slength_concatStorage_asm_0 / 32 (MaverickToken.sol#280)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (MaverickToken.sol#246)' in BytesLib.concatStorage(byte
s,bytes) (MaverickToken.sol#176-311) potentially used before declaration: submod_concatStorage_asm_0 = 32 - slengthmod_concatSto
rage_asm_0 (MaverickToken.sol#289)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatStorage
_asm_0 (MaverickToken.sol#290)
```



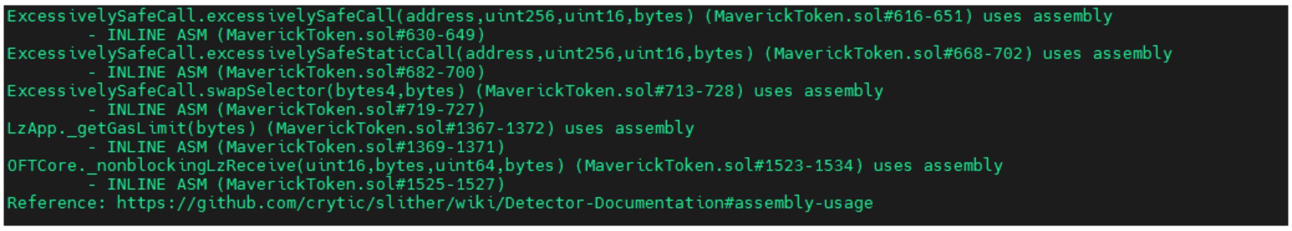
```
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (MaverickToken.sol#246)' in BytesLib.concatStorage(byte
s,bytes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatSto
rage_asm_0 (MaverickToken.sol#290)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (MaverickToken.sol#248)' in BytesLib.concatStorage(bytes,b
ytes) (MaverickToken.sol#176-311) potentially used before declaration: end_concatStorage_asm_0 = _postBytes + mlength_concatStor
age_asm_0 (MaverickToken.sol#291)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (MaverickToken.sol#249)' in BytesLib.concatStorage(bytes,
bytes) (MaverickToken.sol#176-311) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStorage
_asm_0 - 1 (MaverickToken.sol#292)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (MaverickToken.sol#246)' in BytesLib.concatStorage(byte
s,bytes) (MaverickToken.sol#176-311) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStora
ge_asm_0 - 1 (MaverickToken.sol#292)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint2
56)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (MaverickToken.sol#294)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (MaverickToken.sol#249)' in BytesLib.concatStorage(bytes,
bytes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uin
t256)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (MaverickToken.sol#294)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint2
56)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (MaverickToken.sol#294)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (Maver
ickToken.sol#297)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (Ma
verickToken.sol#298)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (Maverick
Token.sol#299)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (MaverickToken.sol#248)' in BytesLib.concatStorage(bytes,b
ytes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (Maveric
kToken.sol#299)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0
 - end_concatStorage_asm_0 (MaverickToken.sol#306)
```



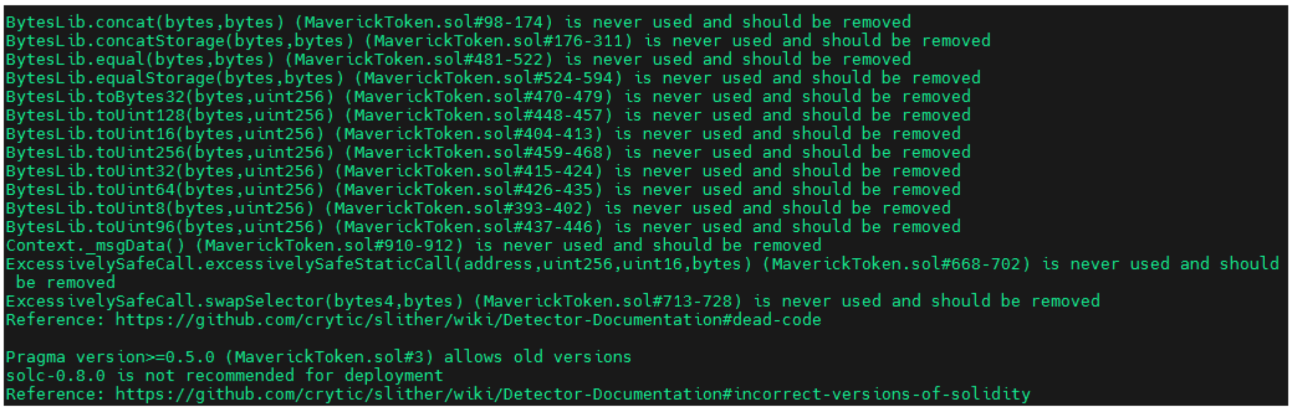
```
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint2
56)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (MaverickToken.sol#308)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint2
56)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (MaverickToken.sol#308)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (MaverickToken.sol#249)' in BytesLib.concatStorage(bytes,
bytes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uin
t256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (MaverickToken.sol#308)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint2
56)(mc_concatStorage_asm_0)) (MaverickToken.sol#303)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint2
56)(mc_concatStorage_asm_0)) (MaverickToken.sol#303)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (MaverickToken.sol#232)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (Maver
ickToken.sol#300)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (MaverickToken.sol#247)' in BytesLib.concatStorage(bytes,by
tes) (MaverickToken.sol#176-311) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (Ma
verickToken.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



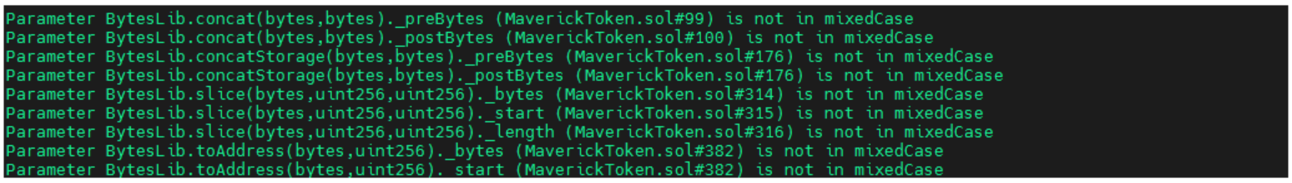
```
Reentrancy in OFTCore._send(address,uint16,bytes,uint256,address,address,bytes) (MaverickToken.sol#1536-1545):
        External calls:
        - _lzSend(_dstChainId,lzPayload,_refundAddress,_zroPaymentAddress,_adapterParams,msg.value) (MaverickToken.sol#1542)
                - lzEndpoint.send{value: _nativeFee}(_dstChainId,trustedRemote,_payload,_refundAddress,_zroPaymentAddress,_adapt
erParams) (MaverickToken.sol#1357)
        Event emitted after the call(s):
        - SendToChain(_dstChainId,_from,_toAddress,amount) (MaverickToken.sol#1544)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BytesLib.concat(bytes,bytes) (MaverickToken.sol#98-174) uses assembly
        - INLINE ASM (MaverickToken.sol#108-171)
BytesLib.concatStorage(bytes,bytes) (MaverickToken.sol#176-311) uses assembly
        - INLINE ASM (MaverickToken.sol#177-310)
BytesLib.slice(bytes,uint256,uint256) (MaverickToken.sol#313-380) uses assembly
        - INLINE ASM (MaverickToken.sol#327-377)
BytesLib.toAddress(bytes,uint256) (MaverickToken.sol#382-391) uses assembly
        - INLINE ASM (MaverickToken.sol#386-388)
BytesLib.toUint8(bytes,uint256) (MaverickToken.sol#393-402) uses assembly
        - INLINE ASM (MaverickToken.sol#397-399)
BytesLib.toUint16(bytes,uint256) (MaverickToken.sol#404-413) uses assembly
        - INLINE ASM (MaverickToken.sol#408-410)
BytesLib.toUint32(bytes,uint256) (MaverickToken.sol#415-424) uses assembly
        - INLINE ASM (MaverickToken.sol#419-421)
BytesLib.toUint64(bytes,uint256) (MaverickToken.sol#426-435) uses assembly
        - INLINE ASM (MaverickToken.sol#430-432)
BytesLib.toUint96(bytes,uint256) (MaverickToken.sol#437-446) uses assembly
        - INLINE ASM (MaverickToken.sol#441-443)
BytesLib.toUint128(bytes,uint256) (MaverickToken.sol#448-457) uses assembly
        - INLINE ASM (MaverickToken.sol#452-454)
BytesLib.toUint256(bytes,uint256) (MaverickToken.sol#459-468) uses assembly
        - INLINE ASM (MaverickToken.sol#463-465)
BytesLib.toBytes32(bytes,uint256) (MaverickToken.sol#470-479) uses assembly
        - INLINE ASM (MaverickToken.sol#474-476)
BytesLib.equal(bytes,bytes) (MaverickToken.sol#481-522) uses assembly
        - INLINE ASM (MaverickToken.sol#484-519)
BytesLib.equalStorage(bytes,bytes) (MaverickToken.sol#524-594) uses assembly
        - INLINE ASM (MaverickToken.sol#534-591)
```



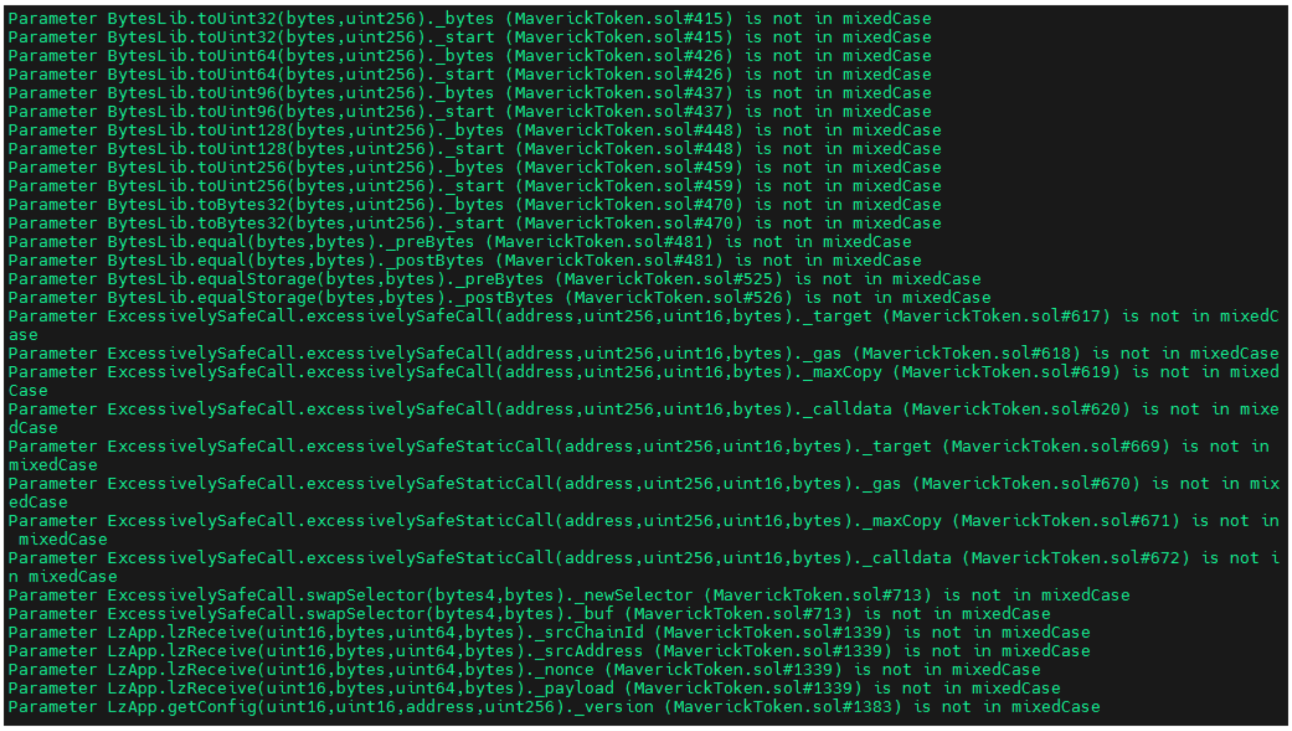
```
ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes) (MaverickToken.sol#616-651) uses assembly
        - INLINE ASM (MaverickToken.sol#630-649)
ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes) (MaverickToken.sol#668-702) uses assembly
        - INLINE ASM (MaverickToken.sol#682-700)
ExcessivelySafeCall.swapSelector(bytes4,bytes) (MaverickToken.sol#713-728) uses assembly
        - INLINE ASM (MaverickToken.sol#719-727)
LzApp._getGasLimit(bytes) (MaverickToken.sol#1367-1372) uses assembly
        - INLINE ASM (MaverickToken.sol#1369-1371)
OFTCore._nonblockingLzReceive(uint16,bytes,uint64,bytes) (MaverickToken.sol#1523-1534) uses assembly
        - INLINE ASM (MaverickToken.sol#1525-1527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



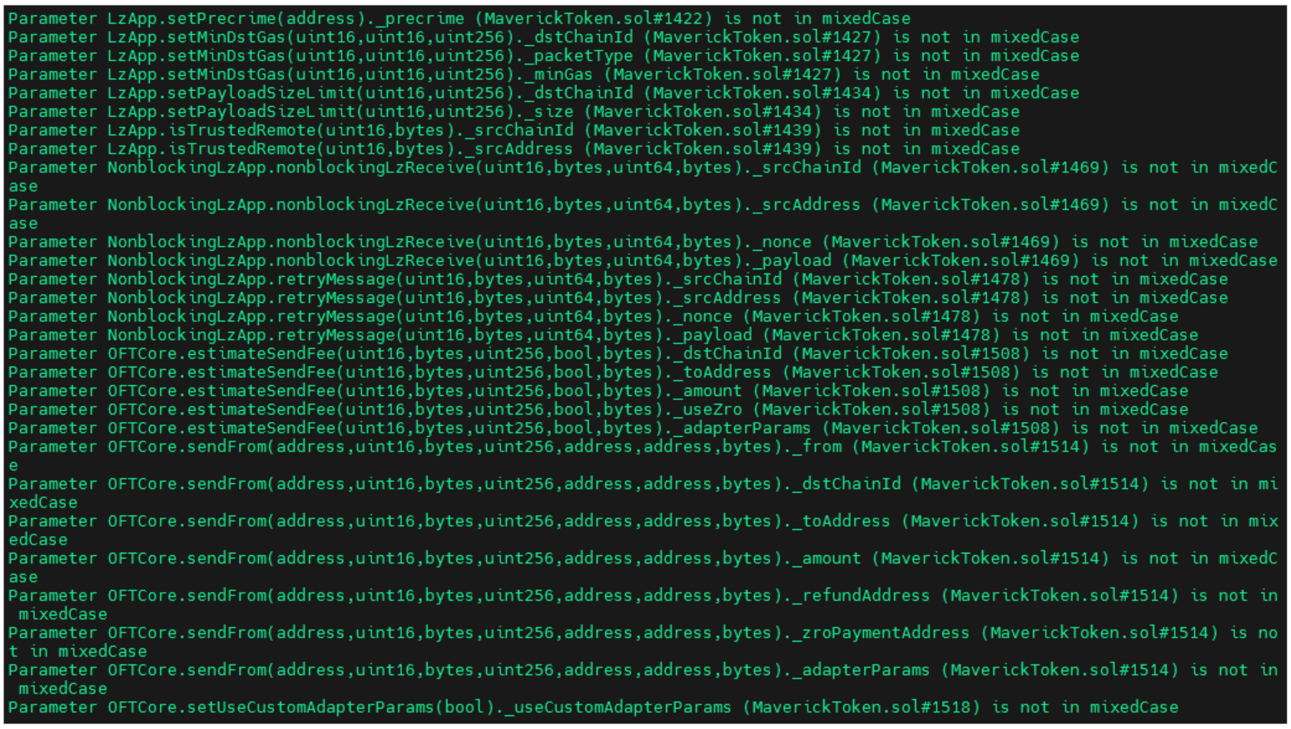
```
BytesLib.concat(bytes,bytes) (MaverickToken.sol#98-174) is never used and should be removed
BytesLib.concatStorage(bytes,bytes) (MaverickToken.sol#176-311) is never used and should be removed
BytesLib.equal(bytes,bytes) (MaverickToken.sol#481-522) is never used and should be removed
BytesLib.equalStorage(bytes,bytes) (MaverickToken.sol#524-594) is never used and should be removed
BytesLib.toBytes32(bytes,uint256) (MaverickToken.sol#470-479) is never used and should be removed
BytesLib.toUint128(bytes,uint256) (MaverickToken.sol#448-457) is never used and should be removed
BytesLib.toUint16(bytes,uint256) (MaverickToken.sol#404-413) is never used and should be removed
BytesLib.toUint256(bytes,uint256) (MaverickToken.sol#459-468) is never used and should be removed
BytesLib.toUint32(bytes,uint256) (MaverickToken.sol#415-424) is never used and should be removed
BytesLib.toUint64(bytes,uint256) (MaverickToken.sol#426-435) is never used and should be removed
BytesLib.toUint8(bytes,uint256) (MaverickToken.sol#393-402) is never used and should be removed
BytesLib.toUint96(bytes,uint256) (MaverickToken.sol#437-446) is never used and should be removed
Context._msgData() (MaverickToken.sol#910-912) is never used and should be removed
ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes) (MaverickToken.sol#668-702) is never used and should
 be removed
ExcessivelySafeCall.swapSelector(bytes4,bytes) (MaverickToken.sol#713-728) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.5.0 (MaverickToken.sol#3) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



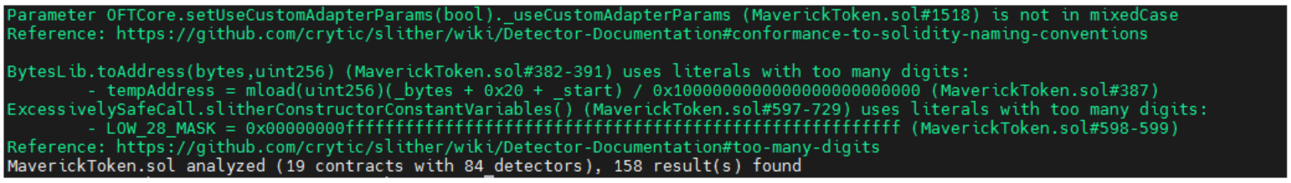
```
Parameter BytesLib.concat(bytes,bytes)._preBytes (MaverickToken.sol#99) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._postBytes (MaverickToken.sol#100) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._preBytes (MaverickToken.sol#176) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._postBytes (MaverickToken.sol#176) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._bytes (MaverickToken.sol#314) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._start (MaverickToken.sol#315) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._length (MaverickToken.sol#316) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._bytes (MaverickToken.sol#382) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._start (MaverickToken.sol#382) is not in mixedCase
```



```
Parameter BytesLib.toUint32(bytes,uint256)._bytes (MaverickToken.sol#415) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._start (MaverickToken.sol#415) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._bytes (MaverickToken.sol#426) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._start (MaverickToken.sol#426) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._bytes (MaverickToken.sol#437) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._start (MaverickToken.sol#437) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._bytes (MaverickToken.sol#448) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._start (MaverickToken.sol#448) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._bytes (MaverickToken.sol#459) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._start (MaverickToken.sol#459) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._bytes (MaverickToken.sol#470) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._start (MaverickToken.sol#470) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._preBytes (MaverickToken.sol#481) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (MaverickToken.sol#481) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (MaverickToken.sol#525) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (MaverickToken.sol#526) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._target (MaverickToken.sol#617) is not in mixedC
ase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._gas (MaverickToken.sol#618) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._maxCopy (MaverickToken.sol#619) is not in mixed
Case
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._calldata (MaverickToken.sol#620) is not in mixe
dCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._target (MaverickToken.sol#669) is not in
mixedCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._gas (MaverickToken.sol#670) is not in mix
edCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._maxCopy (MaverickToken.sol#671) is not in
 mixedCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._calldata (MaverickToken.sol#672) is not i
n mixedCase
Parameter ExcessivelySafeCall.swapSelector(bytes4,bytes)._newSelector (MaverickToken.sol#713) is not in mixedCase
Parameter ExcessivelySafeCall.swapSelector(bytes4,bytes)._buf (MaverickToken.sol#713) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._srcChainId (MaverickToken.sol#1339) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._srcAddress (MaverickToken.sol#1339) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._nonce (MaverickToken.sol#1339) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._payload (MaverickToken.sol#1339) is not in mixedCase
Parameter LzApp.getConfig(uint16,uint16,address,uint256)._version (MaverickToken.sol#1383) is not in mixedCase
```



```
Parameter LzApp.setPrecrime(address)._precrime (MaverickToken.sol#1422) is not in mixedCase
Parameter LzApp.setMinDstGas(uint16,uint16,uint256)._dstChainId (MaverickToken.sol#1427) is not in mixedCase
Parameter LzApp.setMinDstGas(uint16,uint16,uint256)._packetType (MaverickToken.sol#1427) is not in mixedCase
Parameter LzApp.setMinDstGas(uint16,uint16,uint256)._minGas (MaverickToken.sol#1427) is not in mixedCase
Parameter LzApp.setPayloadSizeLimit(uint16,uint256)._dstChainId (MaverickToken.sol#1434) is not in mixedCase
Parameter LzApp.setPayloadSizeLimit(uint16,uint256)._size (MaverickToken.sol#1434) is not in mixedCase
Parameter LzApp.isTrustedRemote(uint16,bytes)._srcChainId (MaverickToken.sol#1439) is not in mixedCase
Parameter LzApp.isTrustedRemote(uint16,bytes)._srcAddress (MaverickToken.sol#1439) is not in mixedCase
Parameter NonblockingLzApp.nonblockingLzReceive(uint16,bytes,uint64,bytes)._srcChainId (MaverickToken.sol#1469) is not in mixedC
ase
Parameter NonblockingLzApp.nonblockingLzReceive(uint16,bytes,uint64,bytes)._srcAddress (MaverickToken.sol#1469) is not in mixedC
ase
Parameter NonblockingLzApp.nonblockingLzReceive(uint16,bytes,uint64,bytes)._nonce (MaverickToken.sol#1469) is not in mixedCase
Parameter NonblockingLzApp.nonblockingLzReceive(uint16,bytes,uint64,bytes)._payload (MaverickToken.sol#1469) is not in mixedCase
Parameter NonblockingLzApp.retryMessage(uint16,bytes,uint64,bytes)._srcChainId (MaverickToken.sol#1478) is not in mixedCase
Parameter NonblockingLzApp.retryMessage(uint16,bytes,uint64,bytes)._srcAddress (MaverickToken.sol#1478) is not in mixedCase
Parameter NonblockingLzApp.retryMessage(uint16,bytes,uint64,bytes)._nonce (MaverickToken.sol#1478) is not in mixedCase
Parameter NonblockingLzApp.retryMessage(uint16,bytes,uint64,bytes)._payload (MaverickToken.sol#1478) is not in mixedCase
Parameter OFTCore.estimateSendFee(uint16,bytes,uint256,bool,bytes)._dstChainId (MaverickToken.sol#1508) is not in mixedCase
Parameter OFTCore.estimateSendFee(uint16,bytes,uint256,bool,bytes)._toAddress (MaverickToken.sol#1508) is not in mixedCase
Parameter OFTCore.estimateSendFee(uint16,bytes,uint256,bool,bytes)._amount (MaverickToken.sol#1508) is not in mixedCase
Parameter OFTCore.estimateSendFee(uint16,bytes,uint256,bool,bytes)._useZro (MaverickToken.sol#1508) is not in mixedCase
Parameter OFTCore.estimateSendFee(uint16,bytes,uint256,bool,bytes)._adapterParams (MaverickToken.sol#1508) is not in mixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._from (MaverickToken.sol#1514) is not in mixedCas
e
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._dstChainId (MaverickToken.sol#1514) is not in mi
xedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._toAddress (MaverickToken.sol#1514) is not in mix
edCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._amount (MaverickToken.sol#1514) is not in mixedC
ase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._refundAddress (MaverickToken.sol#1514) is not in
 mixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._zroPaymentAddress (MaverickToken.sol#1514) is no
t in mixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._adapterParams (MaverickToken.sol#1514) is not in
 mixedCase
Parameter OFTCore.setUseCustomAdapterParams(bool)._useCustomAdapterParams (MaverickToken.sol#1518) is not in mixedCase
```

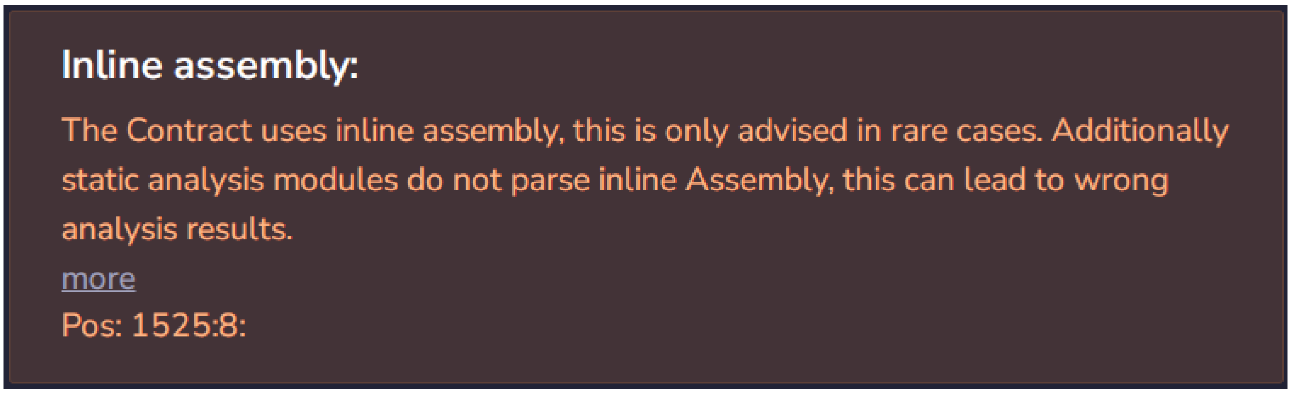


```
Parameter OFTCore.setUseCustomAdapterParams(bool)._useCustomAdapterParams (MaverickToken.sol#1518) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (MaverickToken.sol#382-391) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (MaverickToken.sol#387)
ExcessivelySafeCall.slitherConstructorConstantVariables() (MaverickToken.sol#597-729) uses literals with too many digits:
        - LOW_28_MASK = 0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff (MaverickToken.sol#598-599)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
MaverickToken.sol analyzed (19 contracts with 84 detectors), 158 result(s) found
```

Solidity Static Analysis

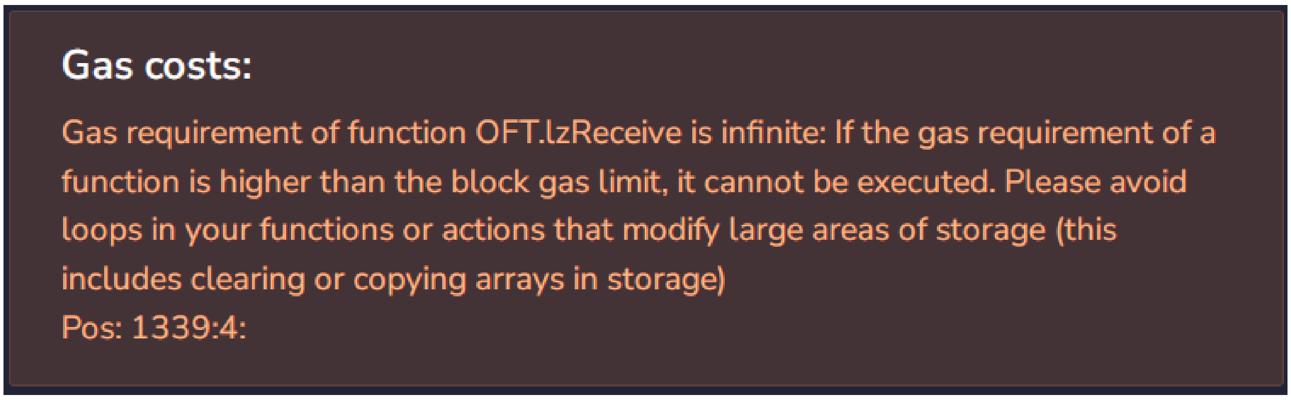
Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

MaverickToken.sol



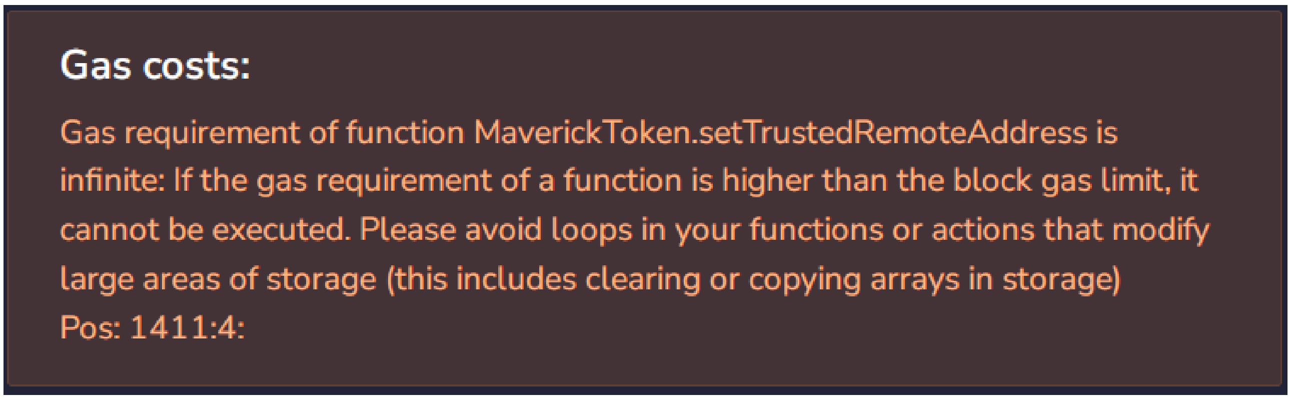
Inline assembly:

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: 1525:8:



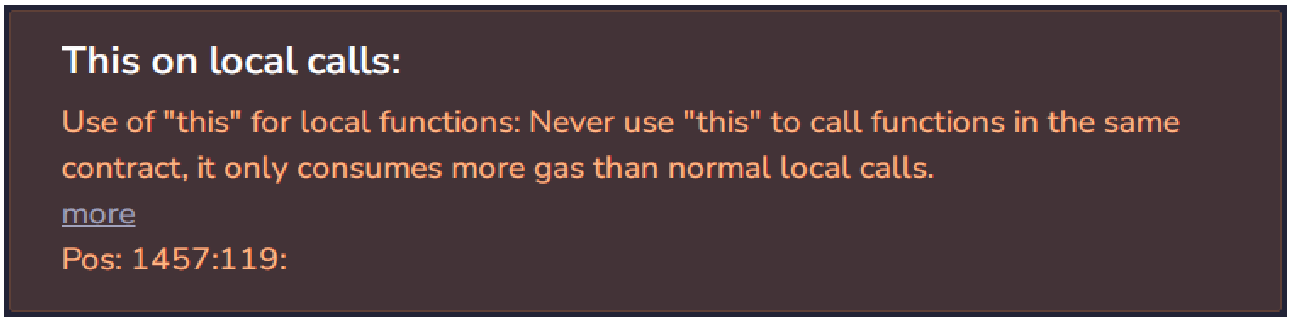
Gas costs:

Gas requirement of function OFT.lzReceive 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: 1339:4:



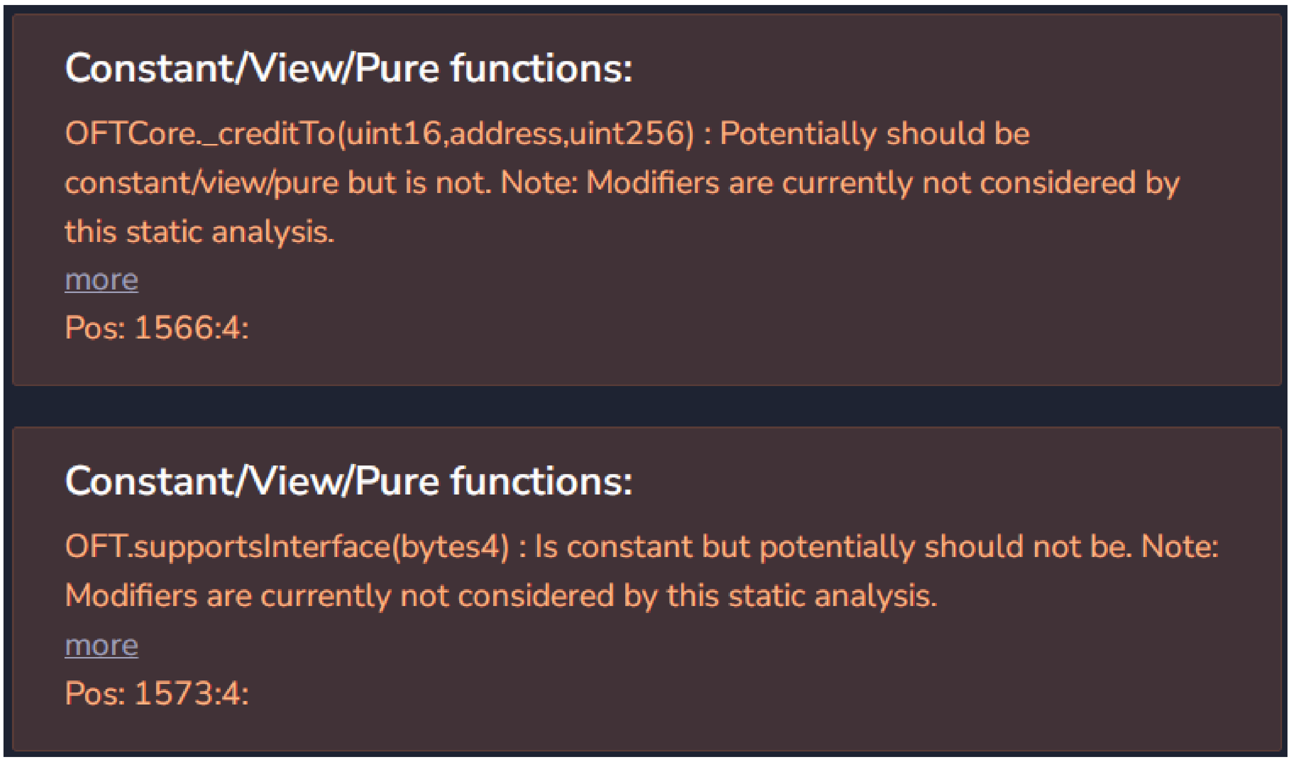
Gas costs:

Gas requirement of function MaverickToken.setTrustedRemoteAddress 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: 1411:4:



This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 1457:119:



Constant/View/Pure functions:

OFTCore._creditTo(uint16,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

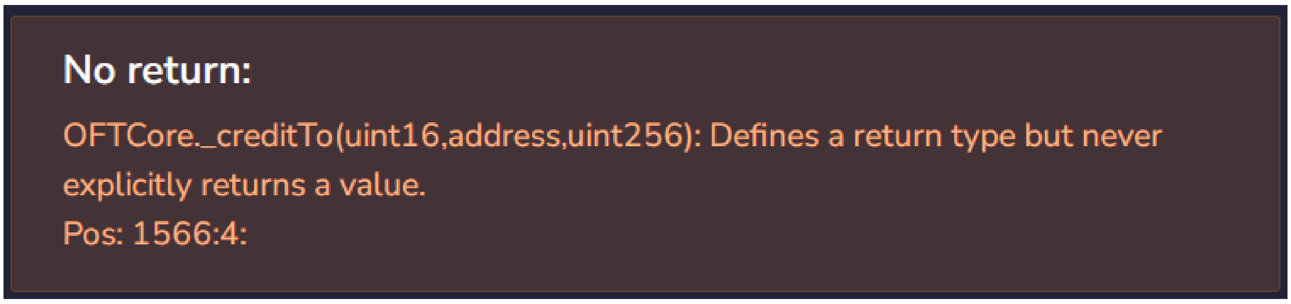
Pos: 1566:4:

Constant/View/Pure functions:

OFT.supportsInterface(bytes4) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

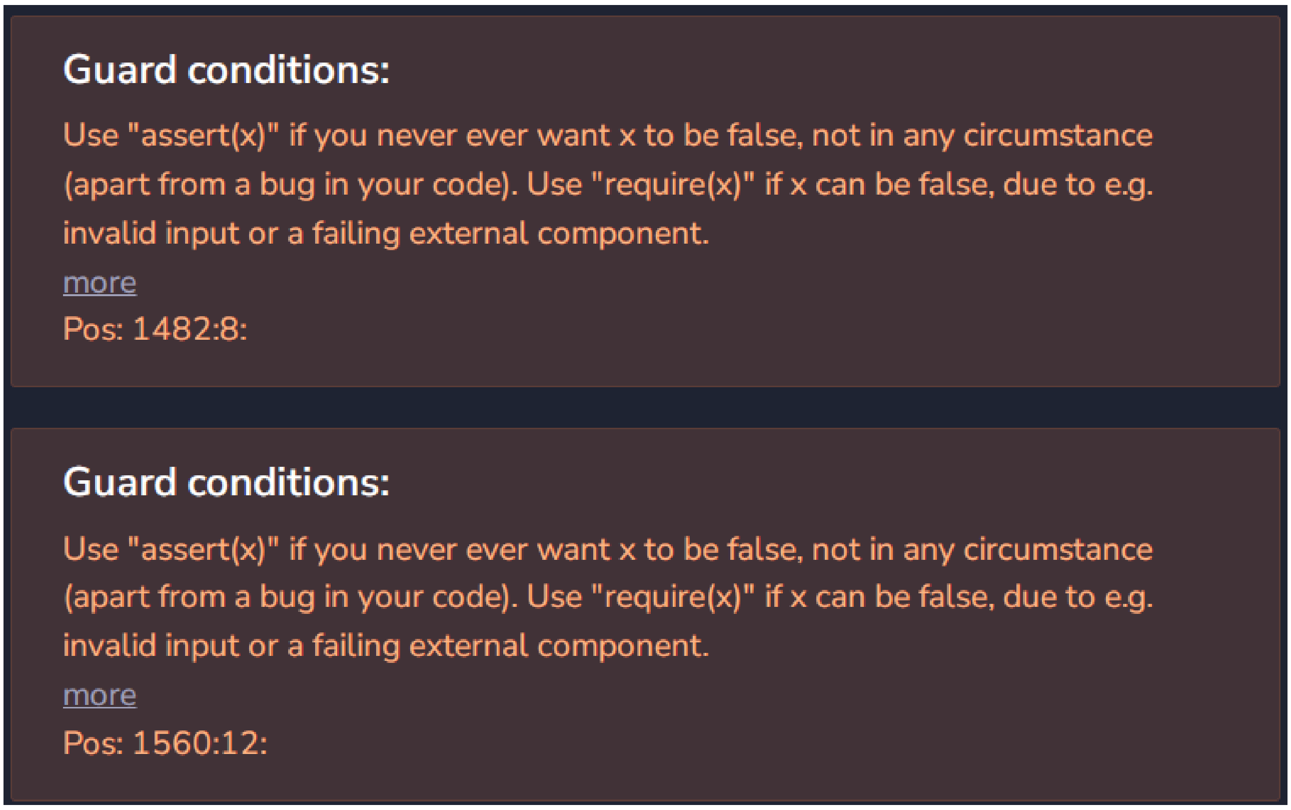
Pos: 1573:4:



No return:

OFTCore._creditTo(uint16,address,uint256): Defines a return type but never explicitly returns a value.

Pos: 1566:4:



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: 1482:8:

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.

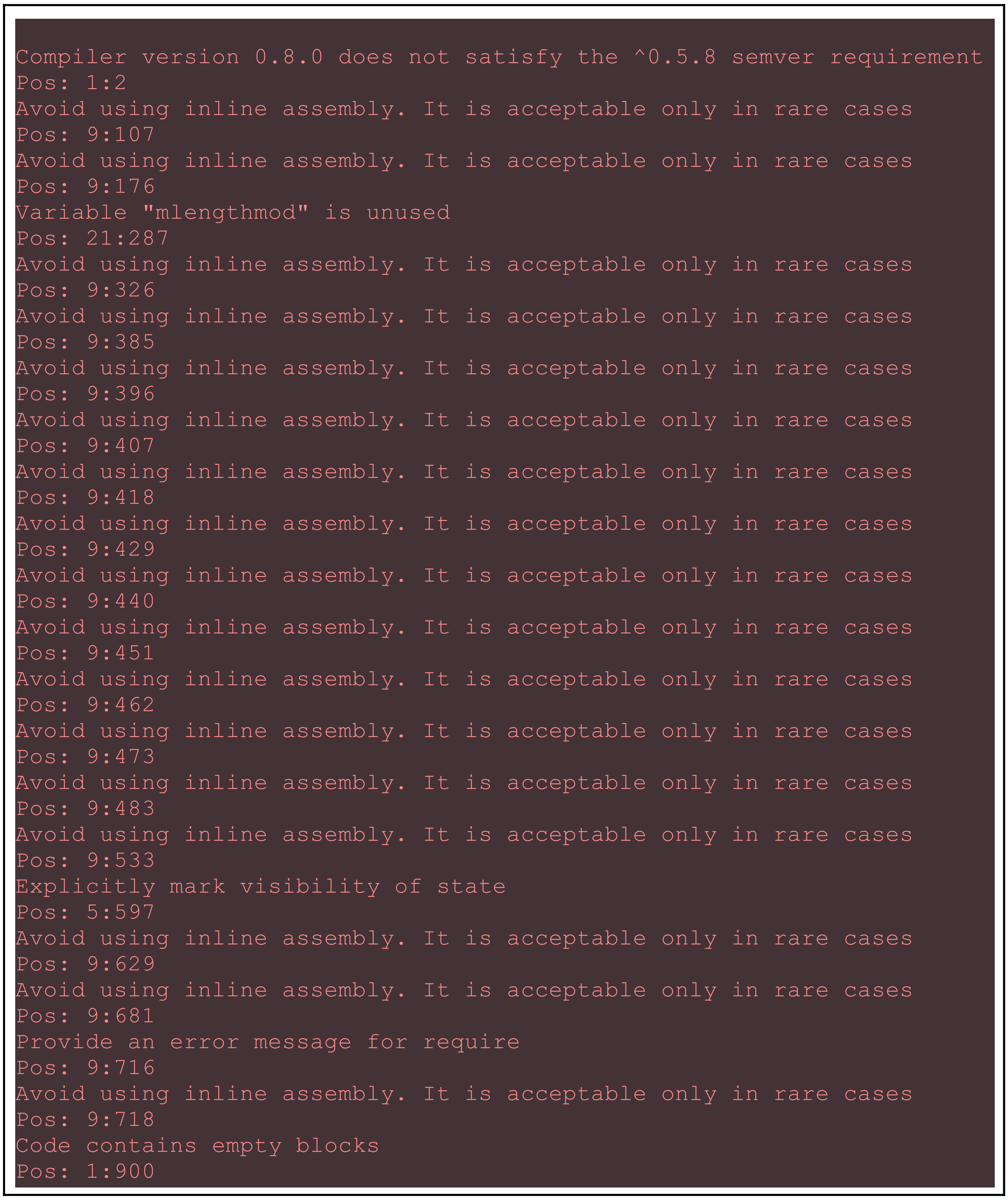
more

Pos: 1560:12:

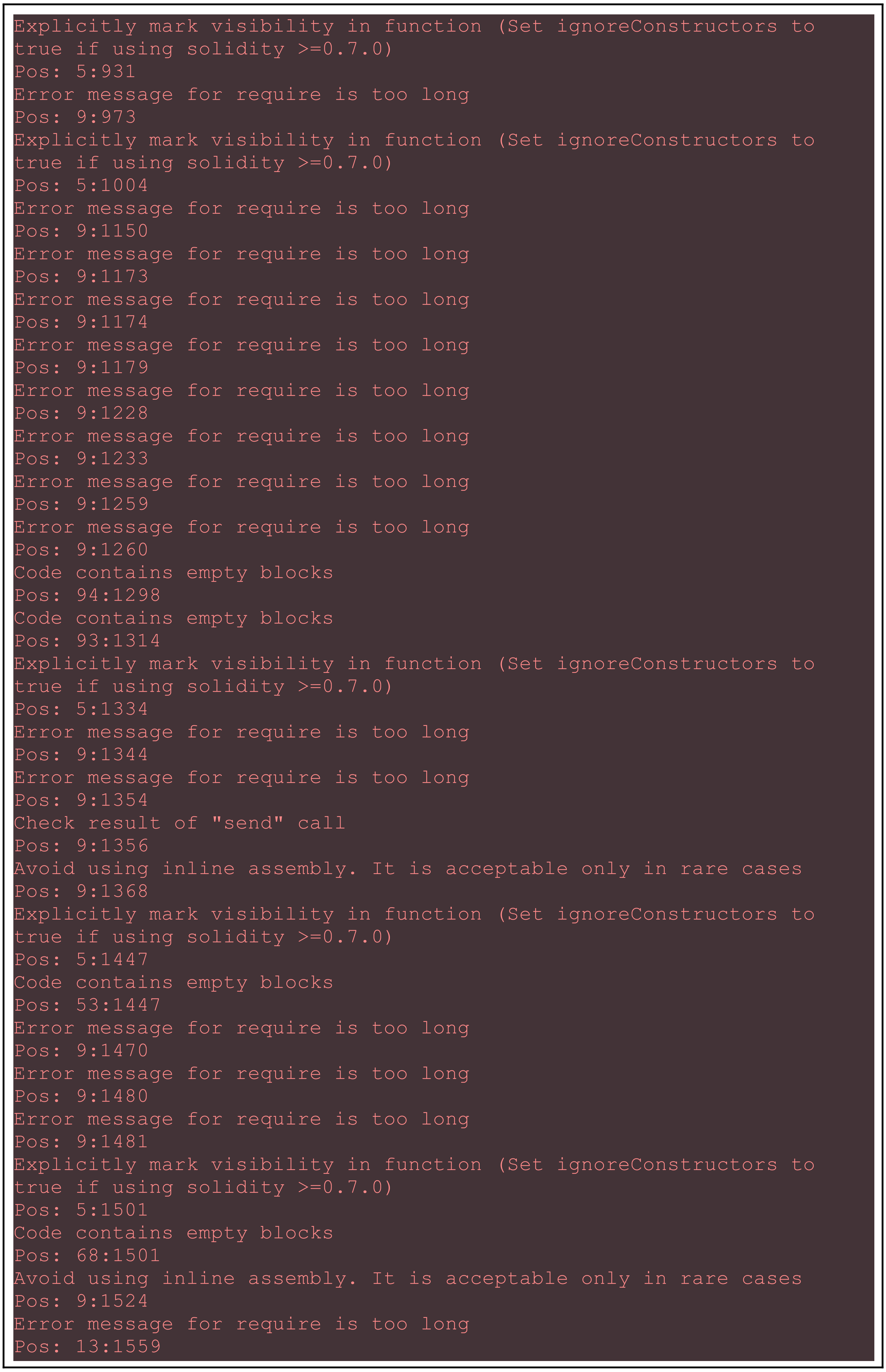
Solhint Linter

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.

MaverickToken.sol



```
Compiler version 0.8.0 does not satisfy the ^0.5.8 semver requirement
Pos: 1:2
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:107
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:176
Variable "mlengthmod" is unused
Pos: 21:287
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:326
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:385
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:396
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:407
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:418
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:429
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:440
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:451
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:462
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:473
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:483
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:533
Explicitly mark visibility of state
Pos: 5:597
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:629
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:681
Provide an error message for require
Pos: 9:716
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:718
Code contains empty blocks
Pos: 1:900
```



```
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:931
Error message for require is too long
Pos: 9:973
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1004
Error message for require is too long
Pos: 9:1150
Error message for require is too long
Pos: 9:1173
Error message for require is too long
Pos: 9:1174
Error message for require is too long
Pos: 9:1179
Error message for require is too long
Pos: 9:1228
Error message for require is too long
Pos: 9:1233
Error message for require is too long
Pos: 9:1259
Error message for require is too long
Pos: 9:1260
Code contains empty blocks
Pos: 94:1298
Code contains empty blocks
Pos: 93:1314
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1334
Error message for require is too long
Pos: 9:1344
Error message for require is too long
Pos: 9:1354
Check result of "send" call
Pos: 9:1356
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:1368
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1447
Code contains empty blocks
Pos: 53:1447
Error message for require is too long
Pos: 9:1470
Error message for require is too long
Pos: 9:1480
Error message for require is too long
Pos: 9:1481
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1501
Code contains empty blocks
Pos: 68:1501
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:1524
Error message for require is too long
Pos: 13:1559
```

```
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1570
Code contains empty blocks
Pos: 125:1570
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1599
```

Software analysis result:

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