Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Tomcat Finance
Website:	https://tomcat.finance
Platform:	Ethereum
Language:	Solidity
Date:	July 17th, 2023

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

EtherAuthority was contracted by the Tomcat Finance team to perform the Security audit of the Tomcat Finance Token, Vesting and token sale smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on July 17th, 2023.

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

- Tomcat is a Convex Maverick Stake MAV for tcMAV, offering full airdrop and boost, and staking earlier for more.
- The Tomcat Finance protocol covers multiple contracts, and all contracts have different functions.
    - TcMav: It is a LayerZero Omni Chain Fungible Token (OFT) and ERC20, a liquid/transferrable receipt token for MAV that is staked into Tomcat Finance.
    - TomcatLaunchVault: Tomcat Finance Vault allows users to stake MAV and receive 1:1 returns.
- The Tomcat Finance protocol contract inherits Ownable, IERC20, SafeERC20 standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope.
- The smart contracts have functions like mint, burn, stake, unstake, etc.

Audit scope

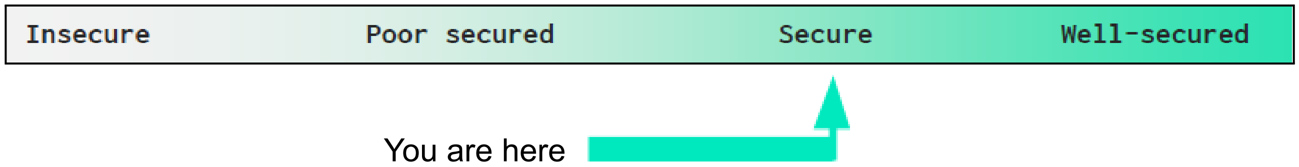
Name	**Code Review and Security Analysis Report for Tomcat Finance Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	TcMav.sol
File 1 MD5 Hash	4A9F0BB3C0FD64E6EA47B455EDD105B7
File 2	TomcatLaunchVault.sol
File 2 MD5 Hash	E1370830C12F616E7E8C14B2B0DB0AC2
Github Commit Hash	02dc14db80e93786138bcfa1aa4b57ad648d9825
Audit Date	July 17th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 TcMav.sol ● Name: Tomcat tcMAV ● Symbol: tcMAV **The owner has control over the following functions:** ● Set whether an account can mint/burn this tcMAV token. ● Creates/Destroys \`amount\` of tcMAV tokens and assigns them to \`account\`, increasing/reducing the total supply. ● Current owners can transfer ownership. ● Owners can renounce ownership.	YES, This is valid.
File 2 TomcatLaunchVault.sol **The owner has control over following functions:** ● Extend the closing time. ● Set the locker contract addresses. ● Current owners can transfer ownership. ● Owners can renounce ownership.	YES, This is valid.

Audit Summary

According to the standard audit assessment, Customer`s solidity 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 critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

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

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	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	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

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

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

Code parts are well commented on smart contracts.

Documentation

We were given a Tomcat Finance smart contract code in the form of a github web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented. 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.

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

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

TcMav.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	supportsInterface	read	Passed	No Issue
9	token	read	Passed	No Issue
10	circulatingSupply	read	Passed	No Issue
11	_debitFrom	internal	Passed	No Issue
12	_creditTo	internal	Passed	No Issue
13	setMinter	external	access only Owner	No Issue
14	mint	external	Minter can mint unlimited tokens	Refer Audit Findings
15	burn	external	Minter can burn anyone’s tokens	Refer Audit Findings
16	onlyMinters	modifier	Passed	No Issue

TomcatLaunchVault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	stake	external	Passed	No Issue
9	unstake	external	Passed	No Issue
10	extendClosingTimestamp	external	access only Owner	No Issue
11	setLocker	external	access only Owner	No Issue
12	lockMav	external	access only Owner	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 in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

(1) Minter can mint unlimited tokens: **TcMav.sol**

In these mentioned functions, users having a minter role can mint unlimited tokens.

Resolution: We suggest adding some limit for tokens to mint. If this is a part of the plan then disregard this issue.

(2) Minter can burn anyone’s token: **TcMav.sol**

Minter can burn any users’ tokens.

Resolution: We suggest changing the code so only token holders can burn their own tokens and not anyone else. Not even a contract creator.

Very Low / Informational / Best practices:

No informational severity vulnerabilities were found in the contract code.

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:

TcMav.sol

- setMinter: Set whether an account can mint/burn this tcMAV token by the owner.
- mint: Creates \`amount\` of tcMAV tokens and assigns them to \`account\`, increasing the total supply by the minters.
- burn: Destroys \`amount\` of tcMAV tokens from \`account\`, reducing the total supply by the minters.

TomcatLaunchVault.sol

- extendClosingTimestamp: Extend closing time can be set by the owner.
- setLocker: The locker contract address can be set by the owner.
- lockMav: Mav can be locked by the owner.

Ownable.sol

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

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

Conclusion

We were given a contract code in the form of a github web link. And we have used all possible tests based on given objects as files. We had observed some Informational severity issues in the smart contracts. but those are not critical. **So, the smart contracts are ready for the mainnet deployment.**

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.

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

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

Our Methodology

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

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

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

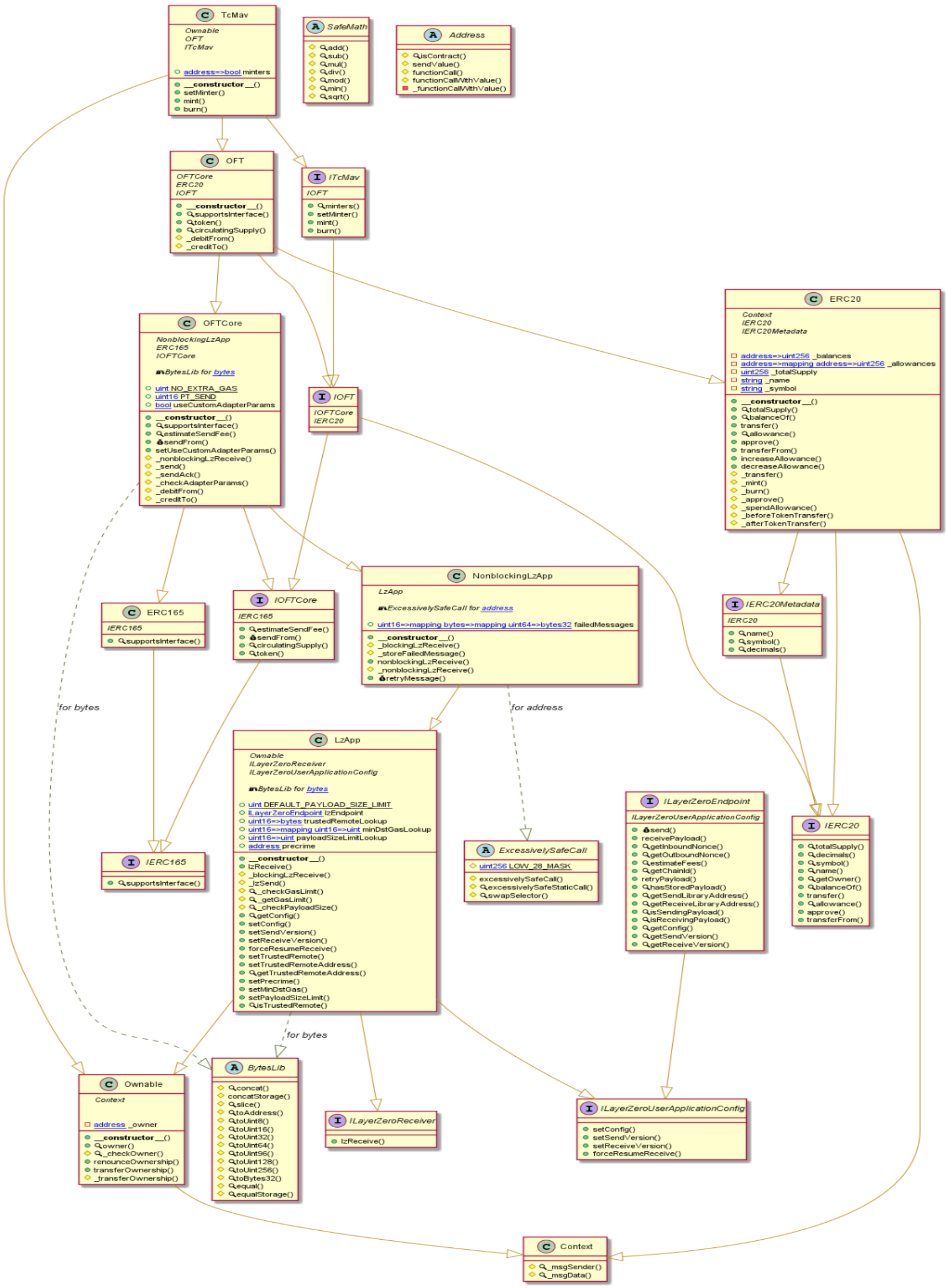
Technical Disclaimer

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

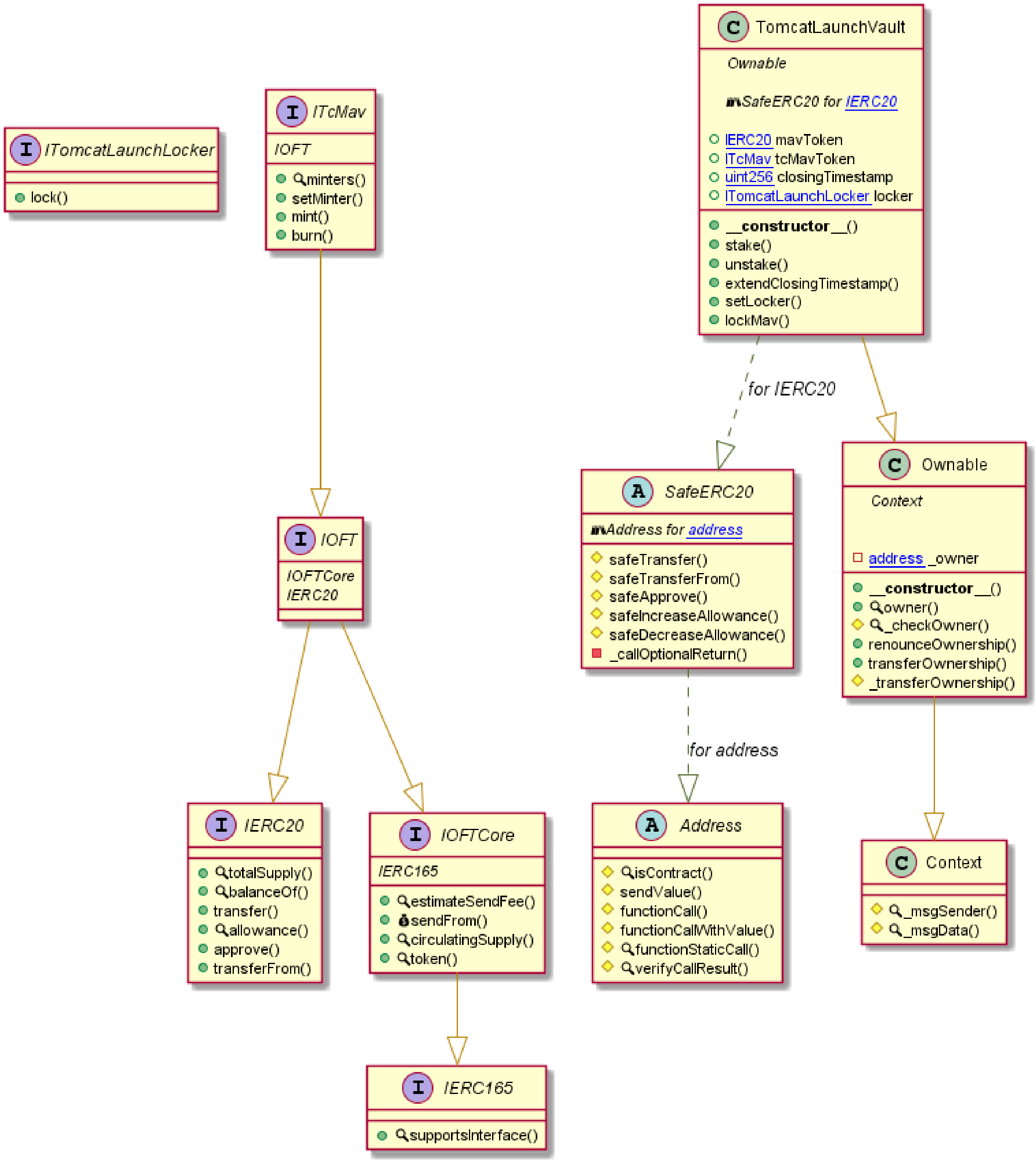
Appendix

Code Flow Diagram - Tomcat Finance

TcMav Diagram

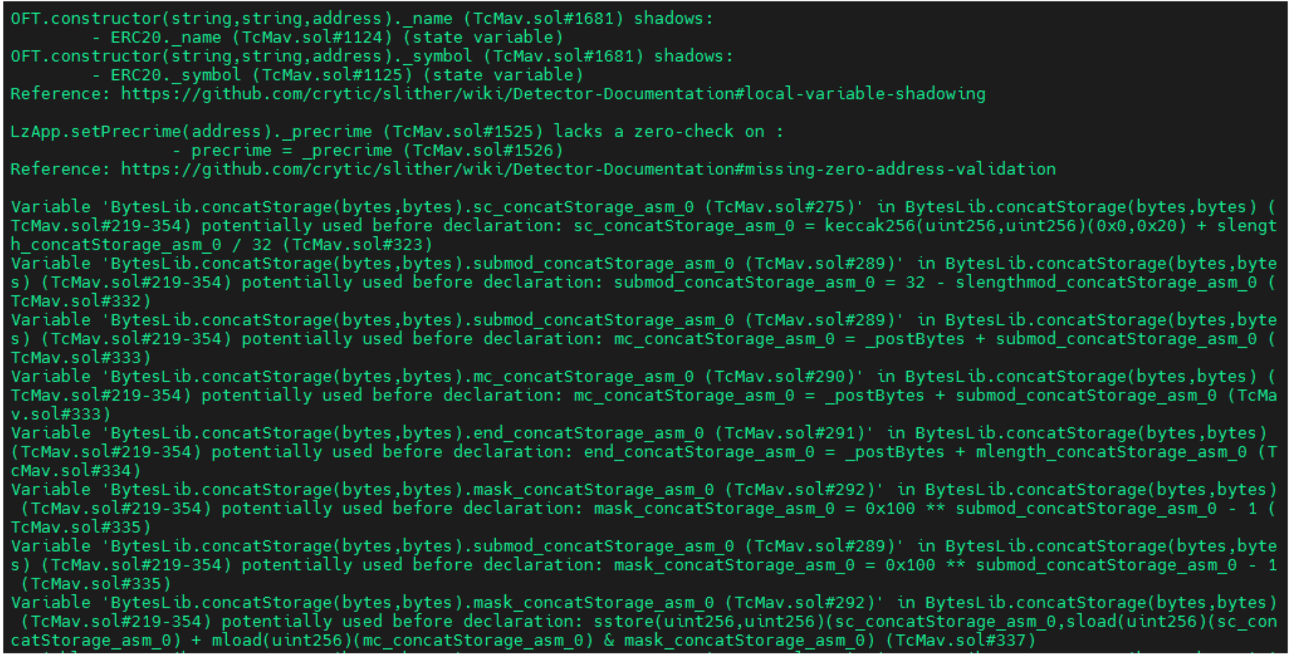


TomcatLaunchVault Diagram

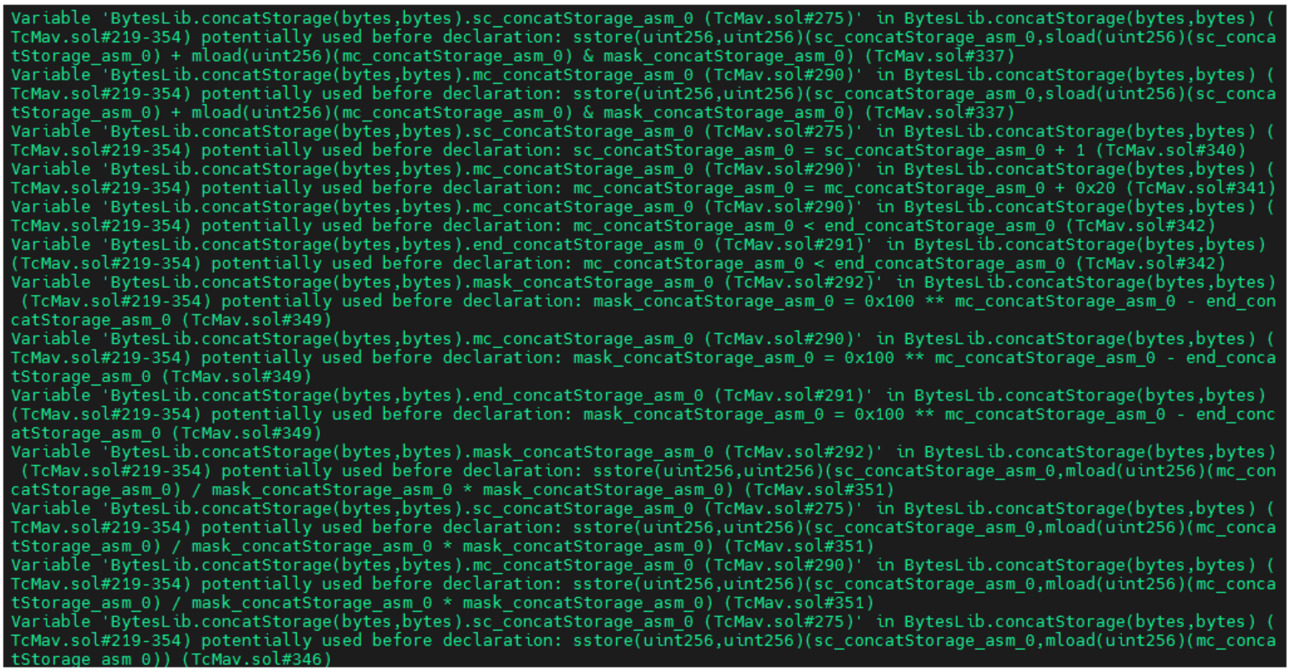


Slither Results Log

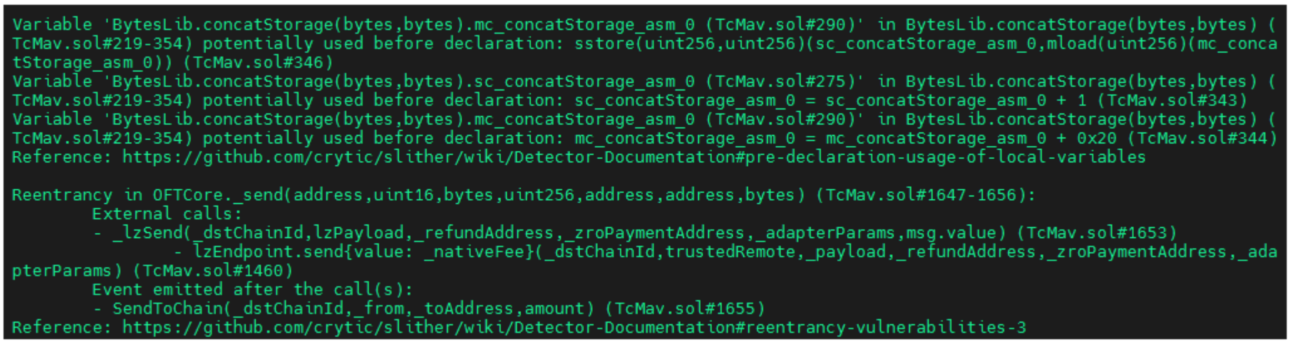
Slither log >> TcMav.sol



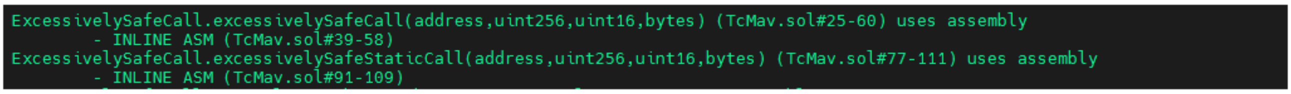
```
OFT.constructor(string,string,address)._name (TcMav.sol#1681) shadows:
        - ERC20._name (TcMav.sol#1124) (state variable)
OFT.constructor(string,string,address)._symbol (TcMav.sol#1681) shadows:
        - ERC20._symbol (TcMav.sol#1125) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

LzApp.setPrecrime(address)._precrime (TcMav.sol#1525) lacks a zero-check on :
                - precrime = _precrime (TcMav.sol#1526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sc_concatStorage_asm_0 = keccak256(uint256,uint256)(0x0,0x20) + slengt
h_concatStorage_asm_0 / 32 (TcMav.sol#323)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (TcMav.sol#289)' in BytesLib.concatStorage(bytes,byte
s) (TcMav.sol#219-354) potentially used before declaration: submod_concatStorage_asm_0 = 32 - slengthmod_concatStorage_asm_0 (
TcMav.sol#332)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (TcMav.sol#289)' in BytesLib.concatStorage(bytes,byte
s) (TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatStorage_asm_0 (
TcMav.sol#333)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatStorage_asm_0 (TcMa
v.sol#333)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (TcMav.sol#291)' in BytesLib.concatStorage(bytes,bytes)
(TcMav.sol#219-354) potentially used before declaration: end_concatStorage_asm_0 = _postBytes + mlength_concatStorage_asm_0 (T
cMav.sol#334)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (TcMav.sol#292)' in BytesLib.concatStorage(bytes,bytes)
 (TcMav.sol#219-354) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStorage_asm_0 - 1 (
TcMav.sol#335)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (TcMav.sol#289)' in BytesLib.concatStorage(bytes,byte
s) (TcMav.sol#219-354) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStorage_asm_0 - 1
 (TcMav.sol#335)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (TcMav.sol#292)' in BytesLib.concatStorage(bytes,bytes)
 (TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_con
catStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (TcMav.sol#337)
```



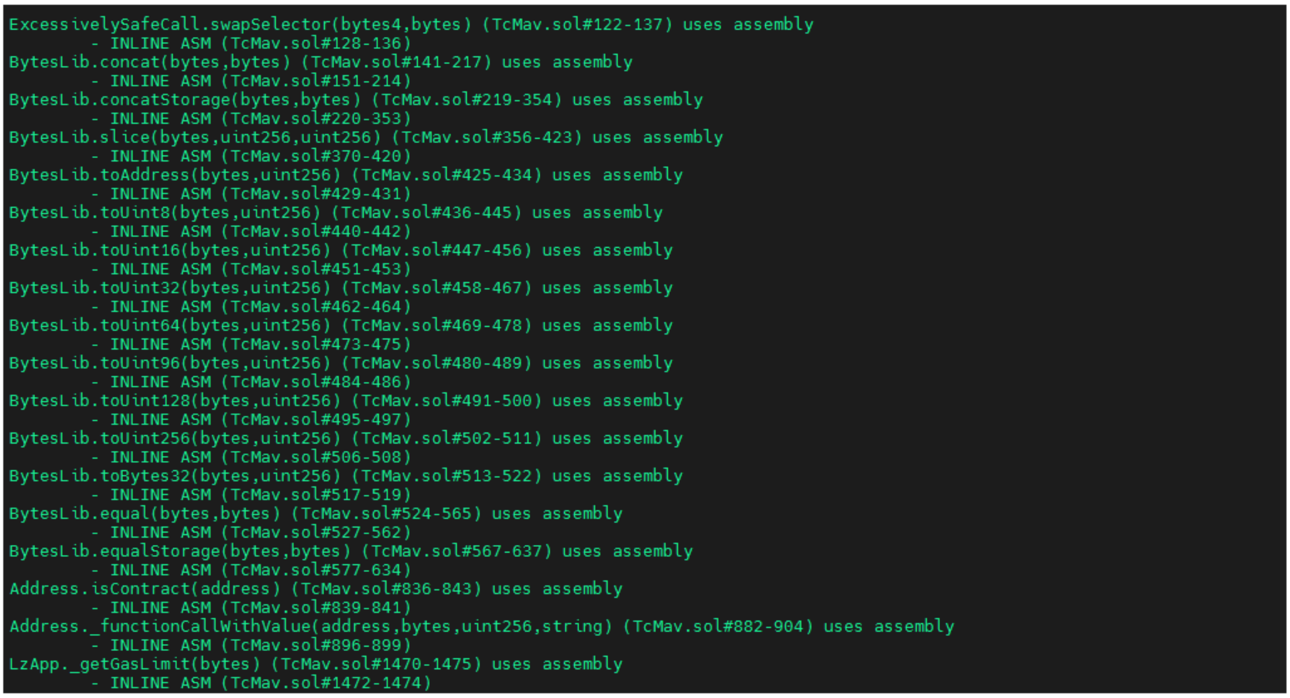
```
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_conca
tStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (TcMav.sol#337)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_conca
tStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (TcMav.sol#337)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (TcMav.sol#340)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (TcMav.sol#341)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (TcMav.sol#342)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (TcMav.sol#291)' in BytesLib.concatStorage(bytes,bytes)
(TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (TcMav.sol#342)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (TcMav.sol#292)' in BytesLib.concatStorage(bytes,bytes)
 (TcMav.sol#219-354) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_con
catStorage_asm_0 (TcMav.sol#349)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_conca
tStorage_asm_0 (TcMav.sol#349)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (TcMav.sol#291)' in BytesLib.concatStorage(bytes,bytes)
(TcMav.sol#219-354) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_conc
atStorage_asm_0 (TcMav.sol#349)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (TcMav.sol#292)' in BytesLib.concatStorage(bytes,bytes)
 (TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_con
catStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (TcMav.sol#351)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_conca
tStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (TcMav.sol#351)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_conca
tStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (TcMav.sol#351)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_conca
tStorage_asm_0)) (TcMav.sol#346)
```



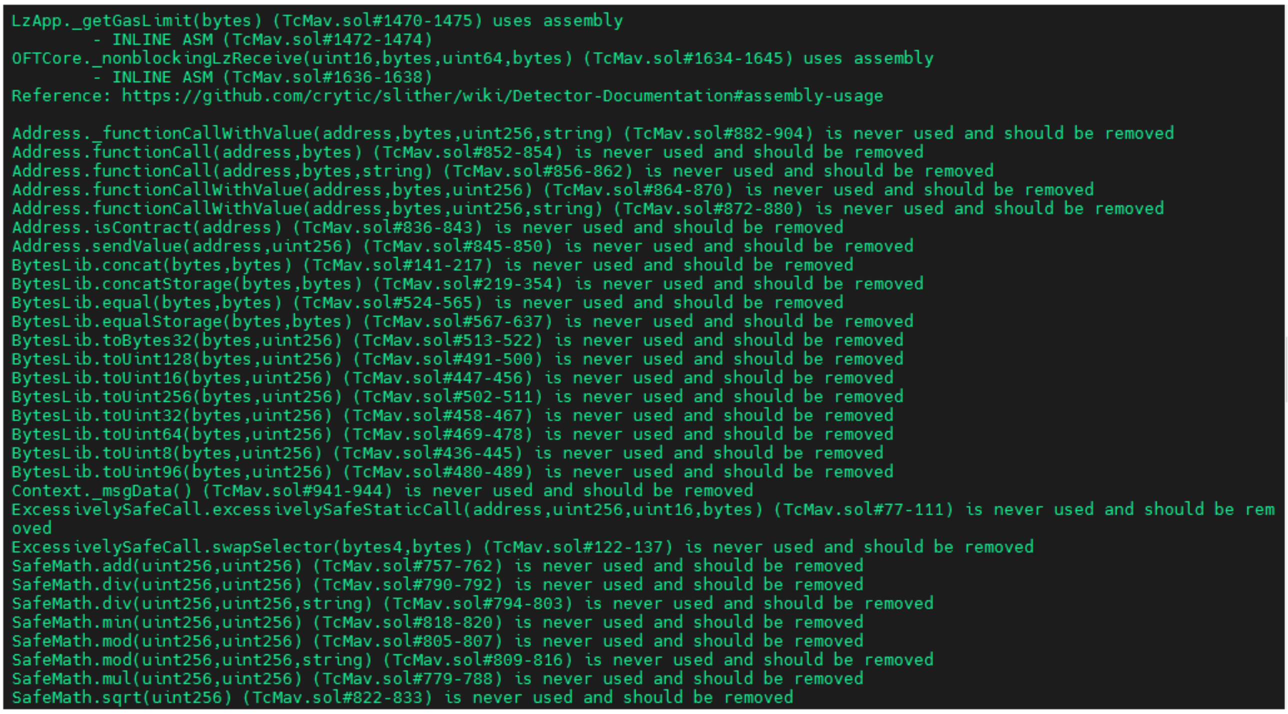
```
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_conca
tStorage_asm_0)) (TcMav.sol#346)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (TcMav.sol#275)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (TcMav.sol#343)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (TcMav.sol#290)' in BytesLib.concatStorage(bytes,bytes) (
TcMav.sol#219-354) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (TcMav.sol#344)
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) (TcMav.sol#1647-1656):
        External calls:
        - _lzSend(_dstChainId,lzPayload,_refundAddress,_zroPaymentAddress,_adapterParams,msg.value) (TcMav.sol#1653)
                - lzEndpoint.send{value: _nativeFee}(_dstChainId,trustedRemote,_payload,_refundAddress,_zroPaymentAddress,_ada
pterParams) (TcMav.sol#1460)
        Event emitted after the call(s):
        - SendToChain(_dstChainId,_from,_toAddress,amount) (TcMav.sol#1655)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



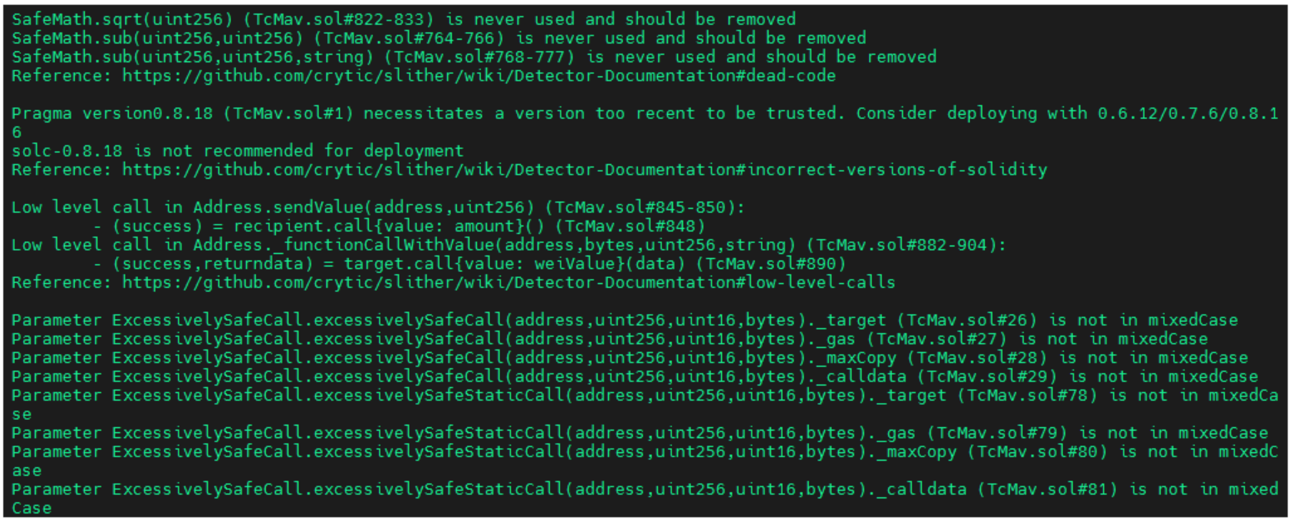
```
ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes) (TcMav.sol#25-60) uses assembly
        - INLINE ASM (TcMav.sol#39-58)
ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes) (TcMav.sol#77-111) uses assembly
        - INLINE ASM (TcMav.sol#91-109)
```



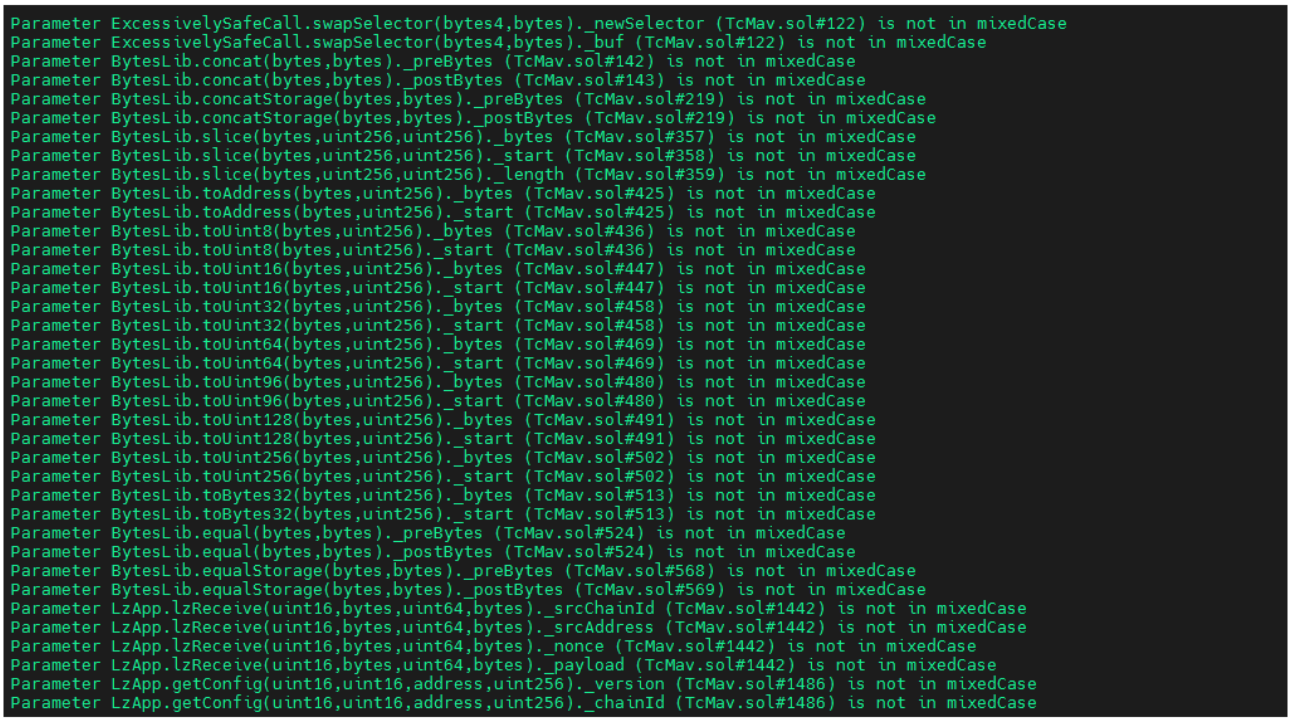
```
ExcessivelySafeCall.swapSelector(bytes4,bytes) (TcMav.sol#122-137) uses assembly
        - INLINE ASM (TcMav.sol#128-136)
BytesLib.concat(bytes,bytes) (TcMav.sol#141-217) uses assembly
        - INLINE ASM (TcMav.sol#151-214)
BytesLib.concatStorage(bytes,bytes) (TcMav.sol#219-354) uses assembly
        - INLINE ASM (TcMav.sol#220-353)
BytesLib.slice(bytes,uint256,uint256) (TcMav.sol#356-423) uses assembly
        - INLINE ASM (TcMav.sol#370-420)
BytesLib.toAddress(bytes,uint256) (TcMav.sol#425-434) uses assembly
        - INLINE ASM (TcMav.sol#429-431)
BytesLib.toUint8(bytes,uint256) (TcMav.sol#436-445) uses assembly
        - INLINE ASM (TcMav.sol#440-442)
BytesLib.toUint16(bytes,uint256) (TcMav.sol#447-456) uses assembly
        - INLINE ASM (TcMav.sol#451-453)
BytesLib.toUint32(bytes,uint256) (TcMav.sol#458-467) uses assembly
        - INLINE ASM (TcMav.sol#462-464)
BytesLib.toUint64(bytes,uint256) (TcMav.sol#469-478) uses assembly
        - INLINE ASM (TcMav.sol#473-475)
BytesLib.toUint96(bytes,uint256) (TcMav.sol#480-489) uses assembly
        - INLINE ASM (TcMav.sol#484-486)
BytesLib.toUint128(bytes,uint256) (TcMav.sol#491-500) uses assembly
        - INLINE ASM (TcMav.sol#495-497)
BytesLib.toUint256(bytes,uint256) (TcMav.sol#502-511) uses assembly
        - INLINE ASM (TcMav.sol#506-508)
BytesLib.toBytes32(bytes,uint256) (TcMav.sol#513-522) uses assembly
        - INLINE ASM (TcMav.sol#517-519)
BytesLib.equal(bytes,bytes) (TcMav.sol#524-565) uses assembly
        - INLINE ASM (TcMav.sol#527-562)
BytesLib.equalStorage(bytes,bytes) (TcMav.sol#567-637) uses assembly
        - INLINE ASM (TcMav.sol#577-634)
Address.isContract(address) (TcMav.sol#836-843) uses assembly
        - INLINE ASM (TcMav.sol#839-841)
Address._functionCallWithValue(address,bytes,uint256,string) (TcMav.sol#882-904) uses assembly
        - INLINE ASM (TcMav.sol#896-899)
LzApp._getGasLimit(bytes) (TcMav.sol#1470-1475) uses assembly
        - INLINE ASM (TcMav.sol#1472-1474)
```



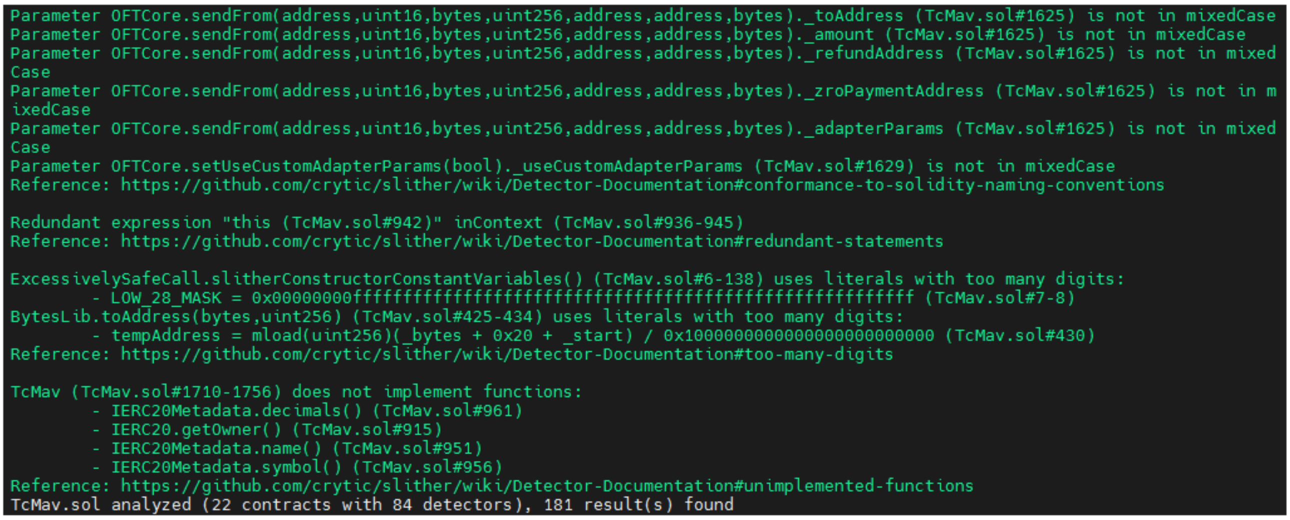
```
LzApp._getGasLimit(bytes) (TcMav.sol#1470-1475) uses assembly
        - INLINE ASM (TcMav.sol#1472-1474)
OFTCore._nonblockingLzReceive(uint16,bytes,uint64,bytes) (TcMav.sol#1634-1645) uses assembly
        - INLINE ASM (TcMav.sol#1636-1638)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (TcMav.sol#882-904) is never used and should be removed
Address.functionCall(address,bytes) (TcMav.sol#852-854) is never used and should be removed
Address.functionCall(address,bytes,string) (TcMav.sol#856-862) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (TcMav.sol#864-870) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (TcMav.sol#872-880) is never used and should be removed
Address.isContract(address) (TcMav.sol#836-843) is never used and should be removed
Address.sendValue(address,uint256) (TcMav.sol#845-850) is never used and should be removed
BytesLib.concat(bytes,bytes) (TcMav.sol#141-217) is never used and should be removed
BytesLib.concatStorage(bytes,bytes) (TcMav.sol#219-354) is never used and should be removed
BytesLib.equal(bytes,bytes) (TcMav.sol#524-565) is never used and should be removed
BytesLib.equalStorage(bytes,bytes) (TcMav.sol#567-637) is never used and should be removed
BytesLib.toBytes32(bytes,uint256) (TcMav.sol#513-522) is never used and should be removed
BytesLib.toUint128(bytes,uint256) (TcMav.sol#491-500) is never used and should be removed
BytesLib.toUint16(bytes,uint256) (TcMav.sol#447-456) is never used and should be removed
BytesLib.toUint256(bytes,uint256) (TcMav.sol#502-511) is never used and should be removed
BytesLib.toUint32(bytes,uint256) (TcMav.sol#458-467) is never used and should be removed
BytesLib.toUint64(bytes,uint256) (TcMav.sol#469-478) is never used and should be removed
BytesLib.toUint8(bytes,uint256) (TcMav.sol#436-445) is never used and should be removed
BytesLib.toUint96(bytes,uint256) (TcMav.sol#480-489) is never used and should be removed
Context._msgData() (TcMav.sol#941-944) is never used and should be removed
ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes) (TcMav.sol#77-111) is never used and should be rem
oved
ExcessivelySafeCall.swapSelector(bytes4,bytes) (TcMav.sol#122-137) is never used and should be removed
SafeMath.add(uint256,uint256) (TcMav.sol#757-762) is never used and should be removed
SafeMath.div(uint256,uint256) (TcMav.sol#790-792) is never used and should be removed
SafeMath.div(uint256,uint256,string) (TcMav.sol#794-803) is never used and should be removed
SafeMath.min(uint256,uint256) (TcMav.sol#818-820) is never used and should be removed
SafeMath.mod(uint256,uint256) (TcMav.sol#805-807) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (TcMav.sol#809-816) is never used and should be removed
SafeMath.mul(uint256,uint256) (TcMav.sol#779-788) is never used and should be removed
SafeMath.sqrt(uint256) (TcMav.sol#822-833) is never used and should be removed
```



```
SafeMath.sqrt(uint256) (TcMav.sol#822-833) is never used and should be removed
SafeMath.sub(uint256,uint256) (TcMav.sol#764-766) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (TcMav.sol#768-777) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.18 (TcMav.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.1
6
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (TcMav.sol#845-850):
        - (success) = recipient.call{value: amount}() (TcMav.sol#848)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (TcMav.sol#882-904):
        - (success,returndata) = target.call{value: weiValue}(data) (TcMav.sol#890)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._target (TcMav.sol#26) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._gas (TcMav.sol#27) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._maxCopy (TcMav.sol#28) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeCall(address,uint256,uint16,bytes)._calldata (TcMav.sol#29) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._target (TcMav.sol#78) is not in mixedCa
se
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._gas (TcMav.sol#79) is not in mixedCase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._maxCopy (TcMav.sol#80) is not in mixedC
ase
Parameter ExcessivelySafeCall.excessivelySafeStaticCall(address,uint256,uint16,bytes)._calldata (TcMav.sol#81) is not in mixed
Case
```

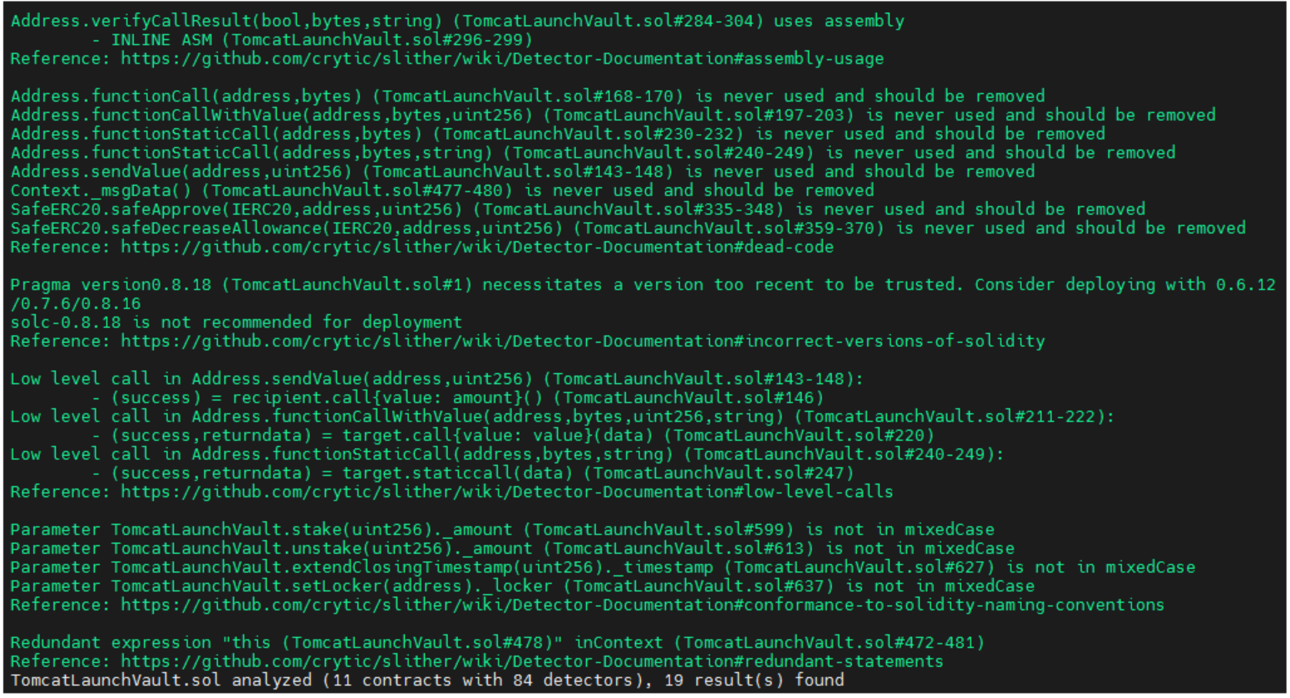


```
Parameter ExcessivelySafeCall.swapSelector(bytes4,bytes)._newSelector (TcMav.sol#122) is not in mixedCase
Parameter ExcessivelySafeCall.swapSelector(bytes4,bytes)._buf (TcMav.sol#122) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._preBytes (TcMav.sol#142) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._postBytes (TcMav.sol#143) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._preBytes (TcMav.sol#219) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._postBytes (TcMav.sol#219) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._bytes (TcMav.sol#357) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._start (TcMav.sol#358) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._length (TcMav.sol#359) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._bytes (TcMav.sol#425) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._start (TcMav.sol#425) is not in mixedCase
Parameter BytesLib.toUint8(bytes,uint256)._bytes (TcMav.sol#436) is not in mixedCase
Parameter BytesLib.toUint8(bytes,uint256)._start (TcMav.sol#436) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._bytes (TcMav.sol#447) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._start (TcMav.sol#447) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._bytes (TcMav.sol#458) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._start (TcMav.sol#458) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._bytes (TcMav.sol#469) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._start (TcMav.sol#469) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._bytes (TcMav.sol#480) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._start (TcMav.sol#480) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._bytes (TcMav.sol#491) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._start (TcMav.sol#491) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._bytes (TcMav.sol#502) is not in mixedCase
Parameter BytesLib.toUint256(bytes,uint256)._start (TcMav.sol#502) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._bytes (TcMav.sol#513) is not in mixedCase
Parameter BytesLib.toBytes32(bytes,uint256)._start (TcMav.sol#513) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._preBytes (TcMav.sol#524) is not in mixedCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (TcMav.sol#524) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (TcMav.sol#568) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (TcMav.sol#569) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._srcChainId (TcMav.sol#1442) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._srcAddress (TcMav.sol#1442) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._nonce (TcMav.sol#1442) is not in mixedCase
Parameter LzApp.lzReceive(uint16,bytes,uint64,bytes)._payload (TcMav.sol#1442) is not in mixedCase
Parameter LzApp.getConfig(uint16,uint16,address,uint256)._version (TcMav.sol#1486) is not in mixedCase
Parameter LzApp.getConfig(uint16,uint16,address,uint256)._chainId (TcMav.sol#1486) is not in mixedCase
```



```
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._toAddress (TcMav.sol#1625) is not in mixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._amount (TcMav.sol#1625) is not in mixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._refundAddress (TcMav.sol#1625) is not in mixed
Case
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._zroPaymentAddress (TcMav.sol#1625) is not in m
ixedCase
Parameter OFTCore.sendFrom(address,uint16,bytes,uint256,address,address,bytes)._adapterParams (TcMav.sol#1625) is not in mixed
Case
Parameter OFTCore.setUseCustomAdapterParams(bool)._useCustomAdapterParams (TcMav.sol#1629) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (TcMav.sol#942)" inContext (TcMav.sol#936-945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

ExcessivelySafeCall.slitherConstructorConstantVariables() (TcMav.sol#6-138) uses literals with too many digits:
        - LOW_28_MASK = 0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff (TcMav.sol#7-8)
BytesLib.toAddress(bytes,uint256) (TcMav.sol#425-434) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (TcMav.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TcMav (TcMav.sol#1710-1756) does not implement functions:
        - IERC20Metadata.decimals() (TcMav.sol#961)
        - IERC20.getOwner() (TcMav.sol#915)
        - IERC20Metadata.name() (TcMav.sol#951)
        - IERC20Metadata.symbol() (TcMav.sol#956)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
TcMav.sol analyzed (22 contracts with 84 detectors), 181 result(s) found
```

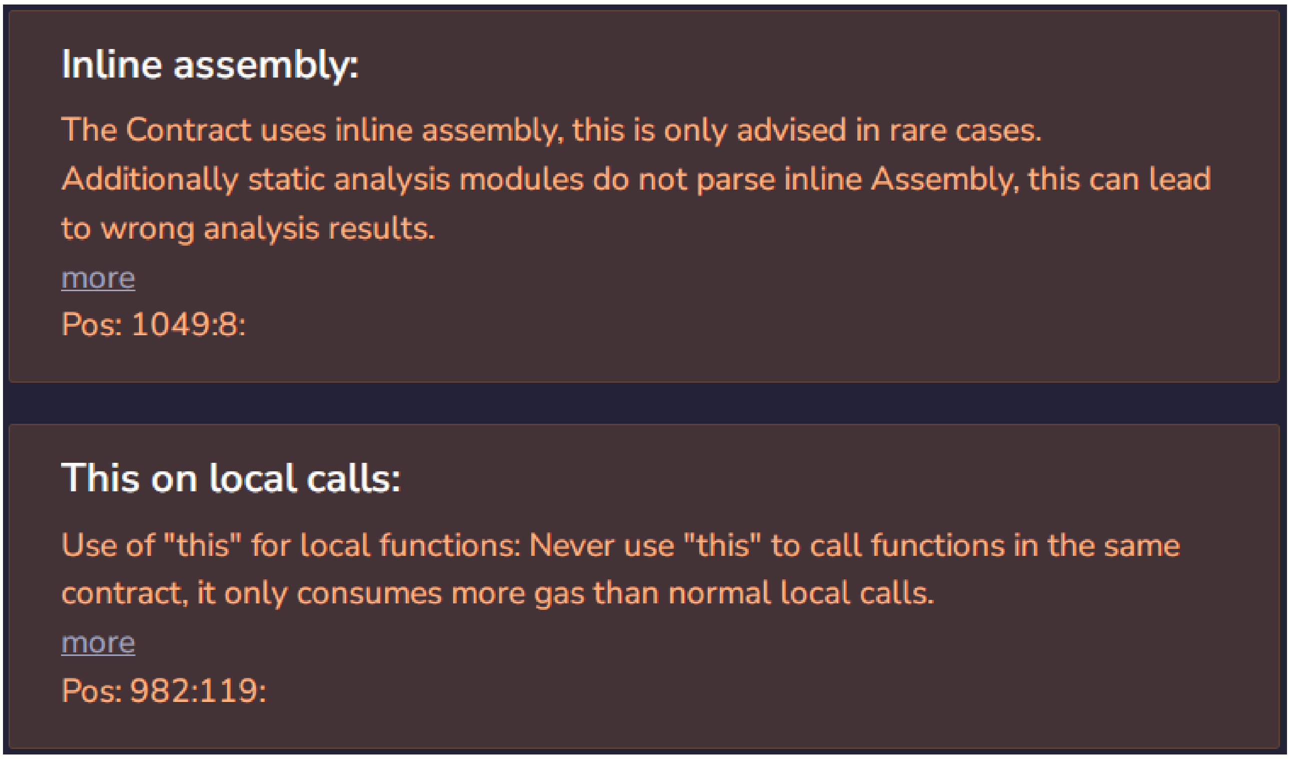
Slither log >> TomcatLaunchVault.sol



```
Address.verifyCallResult(bool,bytes,string) (TomcatLaunchVault.sol#284-304) uses assembly
        - INLINE ASM (TomcatLaunchVault.sol#296-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (TomcatLaunchVault.sol#168-170) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (TomcatLaunchVault.sol#197-203) is never used and should be removed
Address.functionStaticCall(address,bytes) (TomcatLaunchVault.sol#230-232) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (TomcatLaunchVault.sol#240-249) is never used and should be removed
Address.sendValue(address,uint256) (TomcatLaunchVault.sol#143-148) is never used and should be removed
Context._msgData() (TomcatLaunchVault.sol#477-480) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (TomcatLaunchVault.sol#335-348) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (TomcatLaunchVault.sol#359-370) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.18 (TomcatLaunchVault.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12
/0.7.6/0.8.16
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (TomcatLaunchVault.sol#143-148):
        - (success) = recipient.call{value: amount}() (TomcatLaunchVault.sol#146)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (TomcatLaunchVault.sol#211-222):
        - (success,returndata) = target.call{value: value}(data) (TomcatLaunchVault.sol#220)
Low level call in Address.functionStaticCall(address,bytes,string) (TomcatLaunchVault.sol#240-249):
        - (success,returndata) = target.staticcall(data) (TomcatLaunchVault.sol#247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter TomcatLaunchVault.stake(uint256)._amount (TomcatLaunchVault.sol#599) is not in mixedCase
Parameter TomcatLaunchVault.unstake(uint256)._amount (TomcatLaunchVault.sol#613) is not in mixedCase
Parameter TomcatLaunchVault.extendClosingTimestamp(uint256)._timestamp (TomcatLaunchVault.sol#627) is not in mixedCase
Parameter TomcatLaunchVault.setLocker(address)._locker (TomcatLaunchVault.sol#637) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (TomcatLaunchVault.sol#478)" inContext (TomcatLaunchVault.sol#472-481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
TomcatLaunchVault.sol analyzed (11 contracts with 84 detectors), 19 result(s) found
```

Solidity Static Analysis

TcMav.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

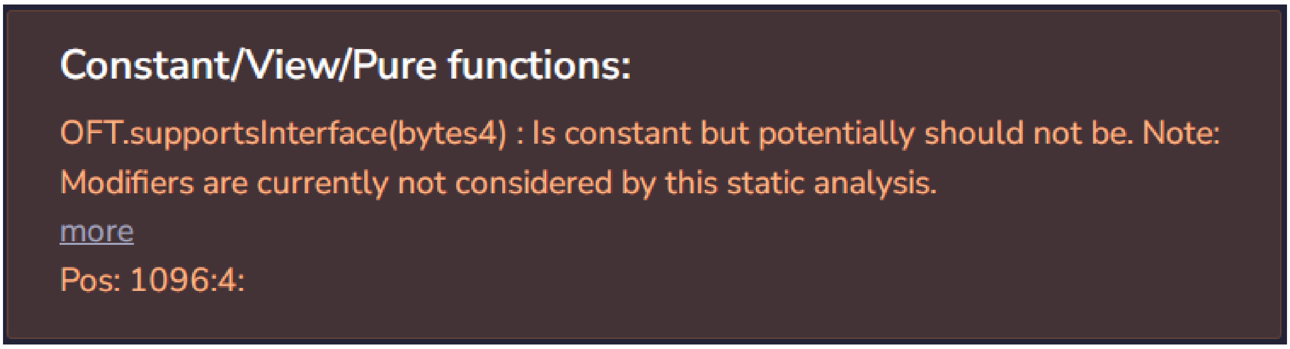
Pos: 1049:8:

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: 982:119:

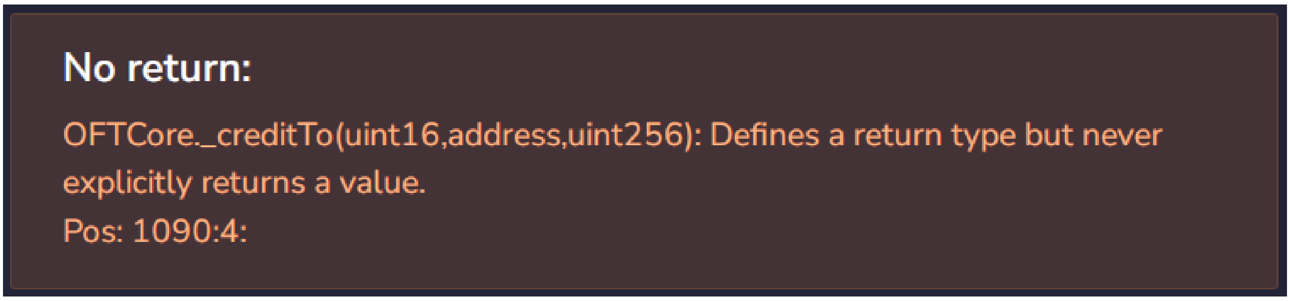


Constant/View/Pure functions:

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

more

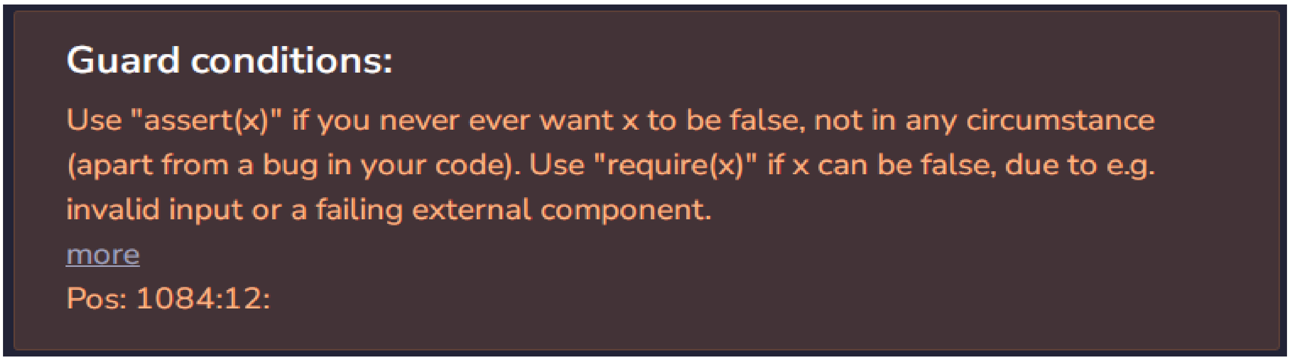
Pos: 1096:4:



No return:

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

Pos: 1090:4:



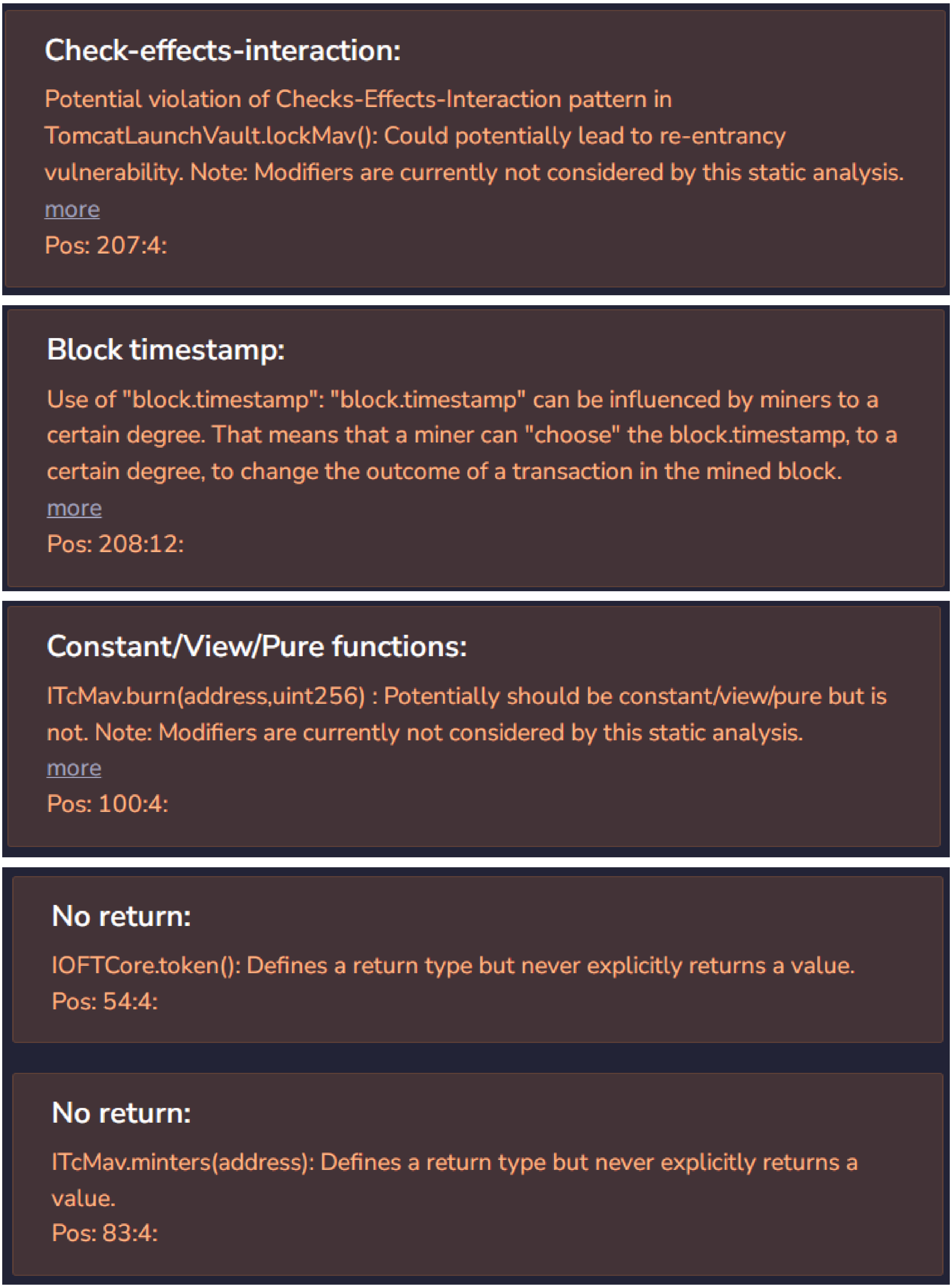
Guard conditions:

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: 1084:12:

TomcatLaunchVault.sol



Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in TomcatLaunchVault.lockMav(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 207:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 208:12:

Constant/View/Pure functions:

ITcMav.burn(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 100:4:

No return:

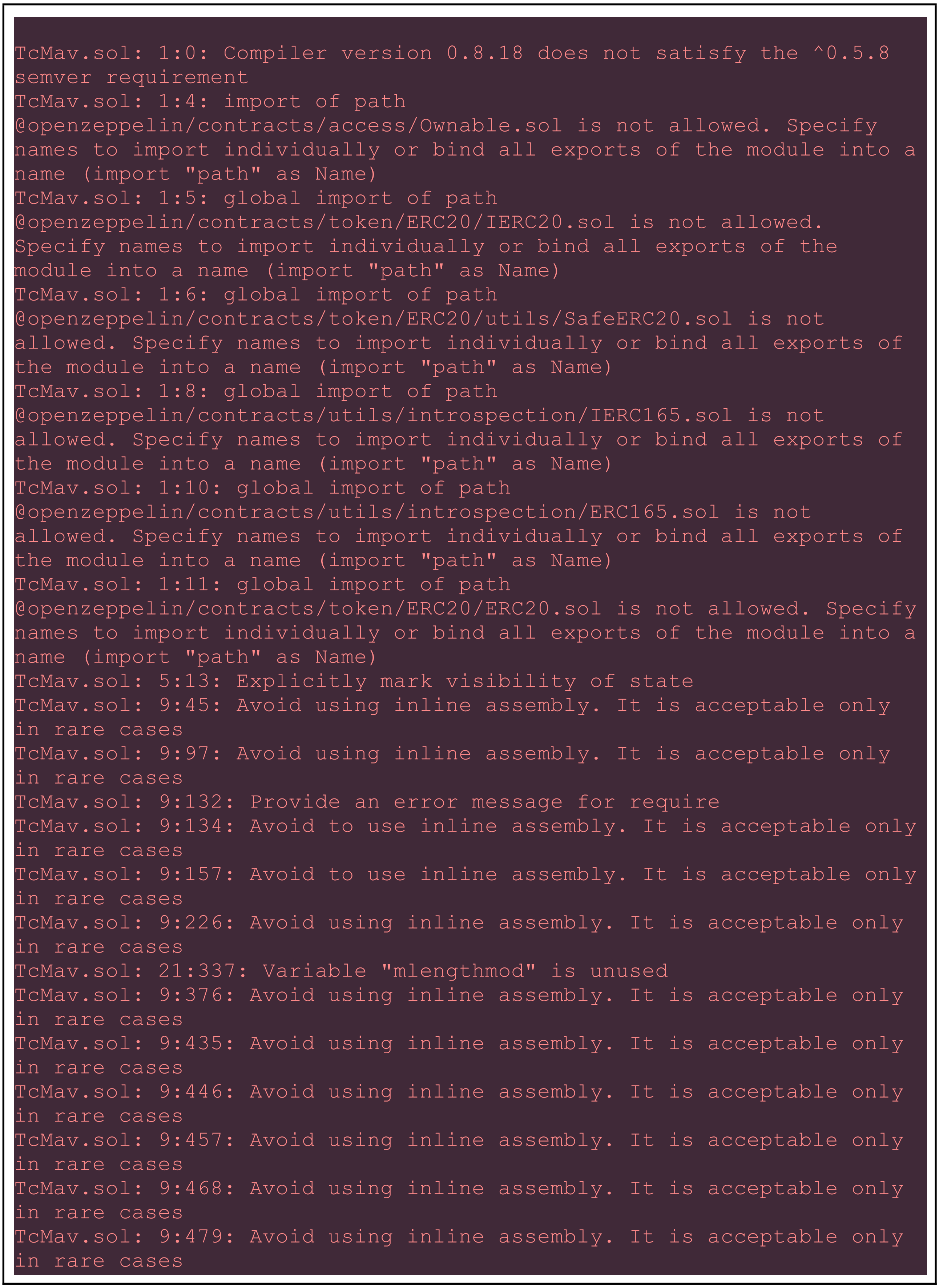
IOFTCore.token(): Defines a return type but never explicitly returns a value.
Pos: 54:4:

No return:

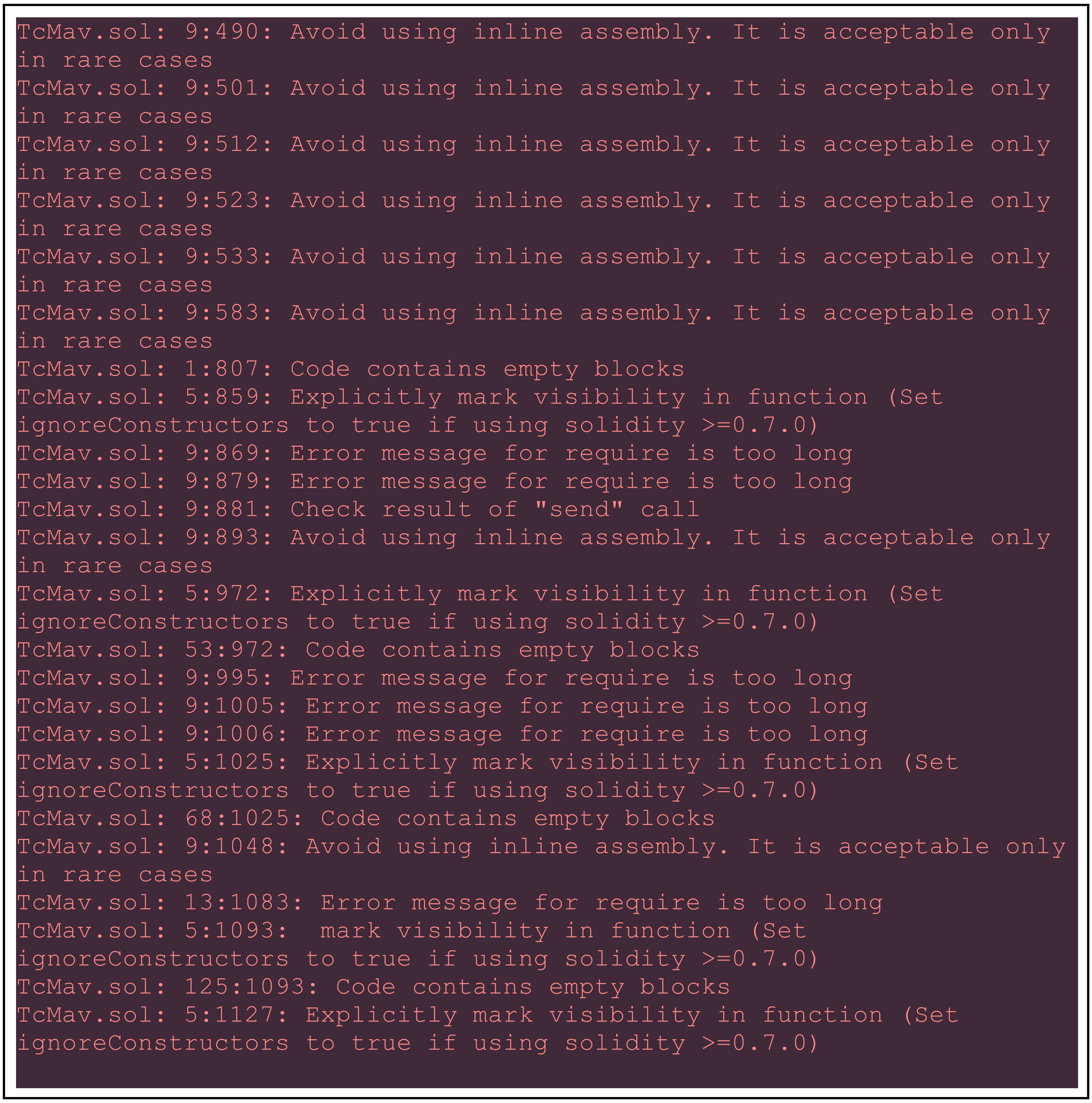
ITcMav.minters(address): Defines a return type but never explicitly returns a value.
Pos: 83:4:

Solhint Linter

TcMav.sol

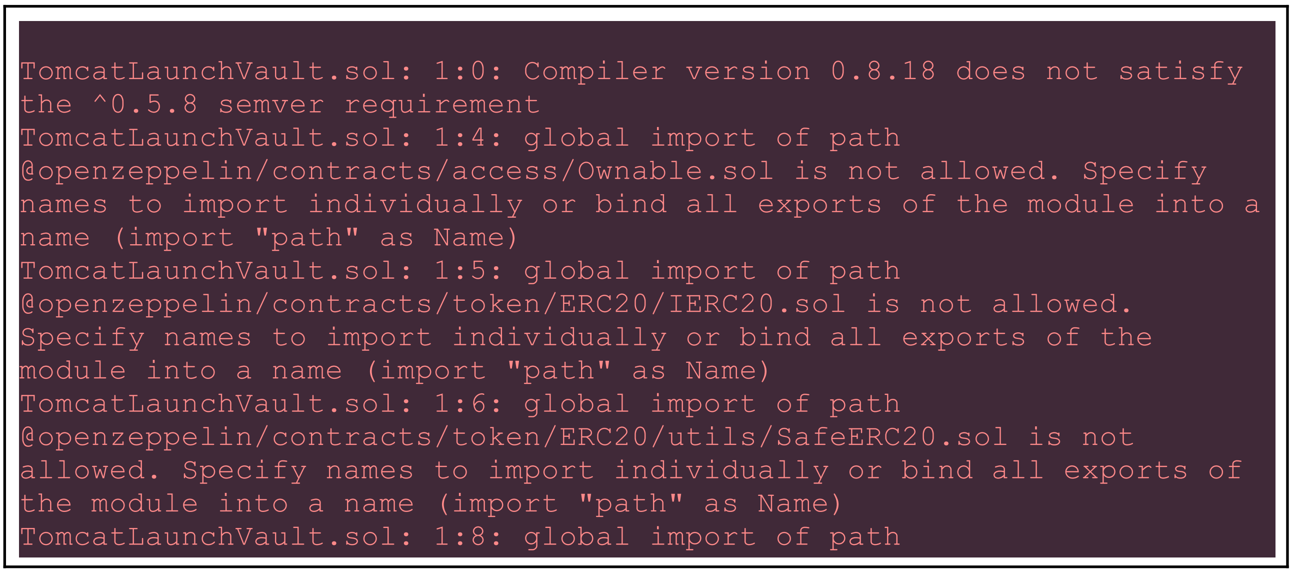


```
TcMav.sol: 1:0: Compiler version 0.8.18 does not satisfy the ^0.5.8
semver requirement
TcMav.sol: 1:4: import of path
@openzeppelin/contracts/access/Ownable.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
TcMav.sol: 1:5: global import of path
@openzeppelin/contracts/token/ERC20/IERC20.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
TcMav.sol: 1:6: global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TcMav.sol: 1:8: global import of path
@openzeppelin/contracts/utils/introspection/IERC165.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TcMav.sol: 1:10: global import of path
@openzeppelin/contracts/utils/introspection/ERC165.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TcMav.sol: 1:11: global import of path
@openzeppelin/contracts/token/ERC20/ERC20.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
TcMav.sol: 5:13: Explicitly mark visibility of state
TcMav.sol: 9:45: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:97: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:132: Provide an error message for require
TcMav.sol: 9:134: Avoid to use inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:157: Avoid to use inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:226: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 21:337: Variable "mlengthmod" is unused
TcMav.sol: 9:376: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:435: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:446: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:457: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:468: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:479: Avoid using inline assembly. It is acceptable only
in rare cases
```

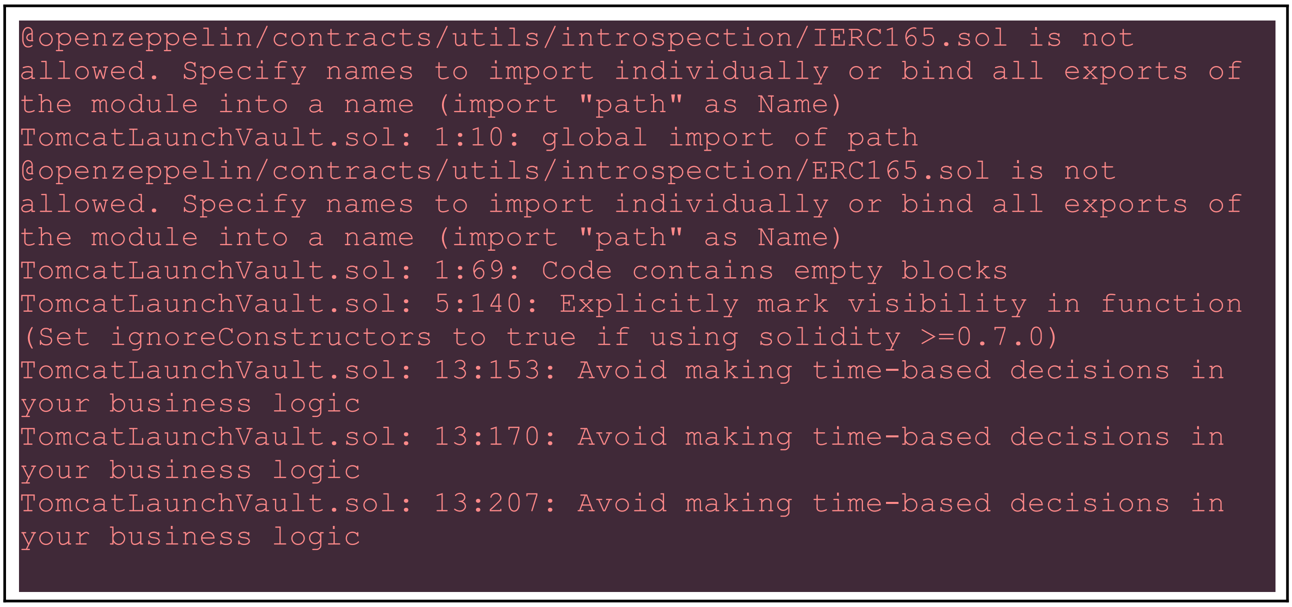


```
TcMav.sol: 9:490: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:501: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:512: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:523: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:533: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 9:583: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 1:807: Code contains empty blocks
TcMav.sol: 5:859: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
TcMav.sol: 9:869: Error message for require is too long
TcMav.sol: 9:879: Error message for require is too long
TcMav.sol: 9:881: Check result of "send" call
TcMav.sol: 9:893: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 5:972: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
TcMav.sol: 53:972: Code contains empty blocks
TcMav.sol: 9:995: Error message for require is too long
TcMav.sol: 9:1005: Error message for require is too long
TcMav.sol: 9:1006: Error message for require is too long
TcMav.sol: 5:1025: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
TcMav.sol: 68:1025: Code contains empty blocks
TcMav.sol: 9:1048: Avoid using inline assembly. It is acceptable only
in rare cases
TcMav.sol: 13:1083: Error message for require is too long
TcMav.sol: 5:1093:  mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
TcMav.sol: 125:1093: Code contains empty blocks
TcMav.sol: 5:1127: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

TomcatLaunchVault.sol



```
TomcatLaunchVault.sol: 1:0: Compiler version 0.8.18 does not satisfy
the ^0.5.8 semver requirement
TomcatLaunchVault.sol: 1:4: global import of path
@openzeppelin/contracts/access/Ownable.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
TomcatLaunchVault.sol: 1:5: global import of path
@openzeppelin/contracts/token/ERC20/IERC20.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
TomcatLaunchVault.sol: 1:6: global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TomcatLaunchVault.sol: 1:8: global import of path
```



```
@openzeppelin/contracts/utils/introspection/IERC165.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TomcatLaunchVault.sol: 1:10: global import of path
@openzeppelin/contracts/utils/introspection/ERC165.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
TomcatLaunchVault.sol: 1:69: Code contains empty blocks
TomcatLaunchVault.sol: 5:140: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
TomcatLaunchVault.sol: 13:153: Avoid making time-based decisions in
your business logic
TomcatLaunchVault.sol: 13:170: Avoid making time-based decisions in
your business logic
TomcatLaunchVault.sol: 13:207: Avoid making time-based decisions in
your business logic
```

Software analysis result:

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

EtherAuthority

Email: audit@EtherAuthority.io