THORSwap Aggregators

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 1st, 2022 - April 17th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/14/2022	István Böhm
0.2	Document Updates	04/17/2022	István Böhm
0.3	Draft Review	04/17/2022	István Böhm
0.4	Draft Review	04/18/2022	Gabi Urrutia
1.0	Remediation Plan	04/22/2022	István Böhm
1.1	Remediation Plan Review	04/23/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
István Böhm	Halborn	Istvan.Bohm@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

THORSwap engaged Halborn to conduct a security audit on their Aggregator smart contracts beginning on April 1st, 2022 and ending on April 17th, 2022. The security assessment was scoped to the Aggregator and vTHOR smart contracts provided in the exchange-contracts GitHub repository thorswap/thorswap-contract-v2.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned one full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified few security risks that were mostly addressed by the THORSwap team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.

2 - May cause temporary impact or loss.

1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

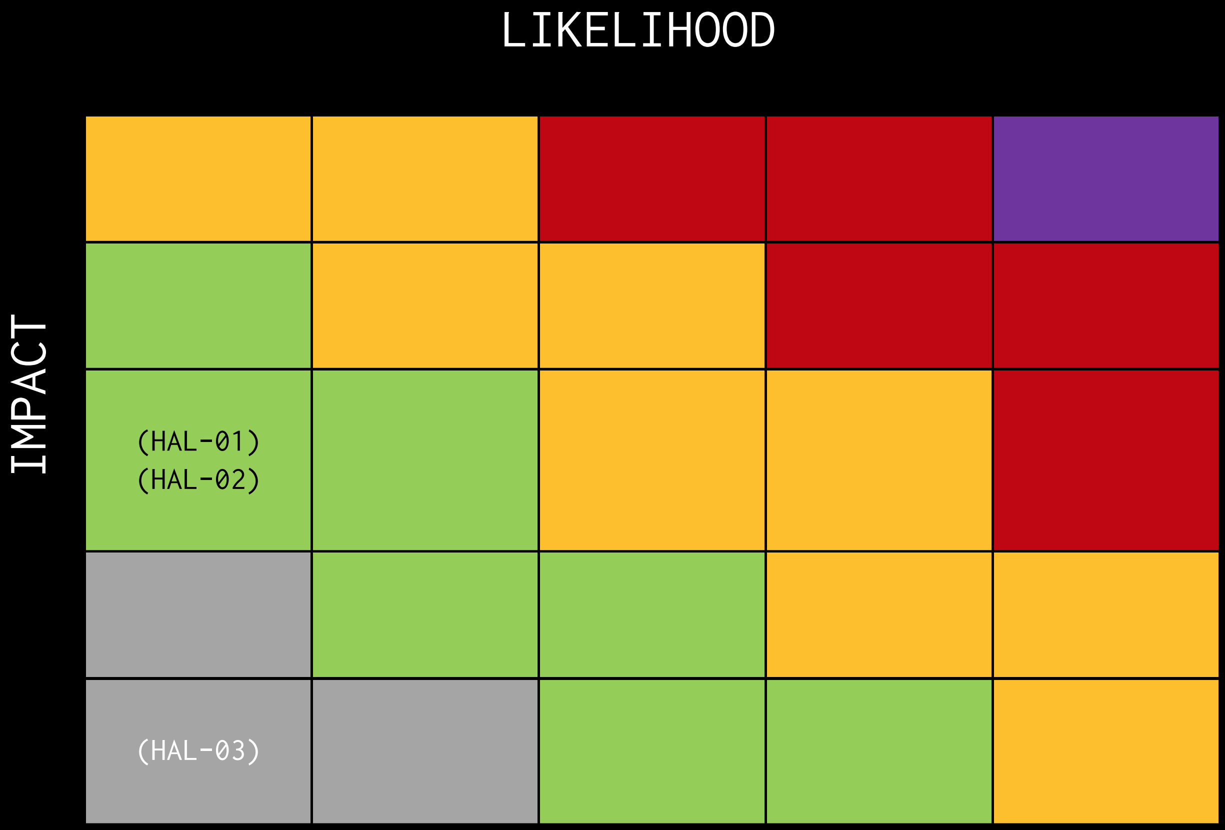
- TSAggregator.sol
- TSAggregator2LegUniswapV2.sol
- TSAggregator2LegUniswapV3.sol
- TSAggregatorGeneric.sol
- TSAggregatorTokenTransferProxy.sol
- TSAggregatorUniswapV2.sol
- TSAggregatorUniswapV3.sol
- vTHOR.sol

Audited Commit ID:
- 6f85c1ebc0e9f8c3ed10c49d4a6db2781c6f5e90

Fixed Commit ID:
- 54fad56917f7055327c7bd2d043259121050923d

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	1



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - INITIAL vTHOR SHARE PRICE MANIPULATION EXPOSURE	Low	RISK ACCEPTED
HAL02 - MISSING RE-ENTRANCY PROTECTION	Low	SOLVED - 04/21/2022
HAL03 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 04/21/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) INITIAL vTHOR SHARE PRICE MANIPULATION EXPOSURE - LOW

Description:

After deployment, the vTHOR contract is initialed without seeding liquidity, exposing the vault to share price manipulation attacks.

Proof of Concept:

1. The vTHOR contract is deployed by the THORSwap team.
2. An attacker finds the vTHOR contract before anyone can deposit their tokens.
3. The attacker deposits 1 token, for which they receive 1 share. This transaction sets the share price at 1 token/share.
4. The attacker then transfers an additional 999 tokens to the vTHOR contract. This transaction increases the share price to 1000 tokens/share.
5. A victim user finds the vTHOR contract and deposits 1500 tokens. Due to the integer rounding, the user only receives 1 share. This transaction modifies the share price to 1250 tokens/share.
6. The attacker then, withdraws from the contract using the redeem function. Depending on the distribution of shares, they will receive 1250 tokens, of which 250 will belong to the victim user.

We note that the impact and likelihood of the vulnerability are low because the attacker has limited opportunity to exploit the issue. However, contract developers must be aware of the initial exposure to prevent potential damage.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

One of the possible solutions to avoid the price manipulation attack is to lock some tokens in the contract and then manually validate the share price before the vTHOR contract is made public.

Remediation Plan:

RISK ACCEPTED: The THORSwap team accept the risk of this finding and will deploy vTHOR and deposit THOR from treasury first to mitigate the issue.

3.2 (HAL-02) MISSING RE-ENTRANCY PROTECTION - LOW

Description:

The vTHOR contract missed the nonReentrant guard in the deposit, mint , withdraw and redeem public functions. Even if the functions follow the check-effects-interactions pattern, we recommend using a mutex to be protected against cross-function reentrancy attacks. By using this lock, an attacker can no longer exploit the function with a recursive call.

Note that the vTHOR contract included a mutex implementation called ReentrancyGuard, which provides a modifier to any function called nonReentrant that guards with a mutex against reentrancy attacks. However, the modifier is not used within the contract.

Code Location:

Listing 1: vTHOR.sol

```
function deposit(uint256 assets, address receiver) public returns
↳ (uint256 shares) {
    // Check for rounding error since we round down in
↳ previewDeposit.
    require((shares = previewDeposit(assets)) != 0, "ZERO_SHARES")
↳ ;
    // Need to transfer before minting or ERC777s could reenter.
    address(_asset).safeTransferFrom(msg.sender, address(this),
↳ assets);
    _mint(receiver, shares);
    emit Deposit(msg.sender, receiver, assets, shares);
}

function mint(uint256 shares, address receiver) public returns (
↳ uint256 assets) {
    assets = previewMint(shares); // No need to check for rounding
↳  error, previewMint rounds up.
    // Need to transfer before minting or ERC777s could reenter.
    address(_asset).safeTransferFrom(msg.sender, address(this),
assets);
    _mint(receiver, shares);
    emit Deposit(msg.sender, receiver, assets, shares);
}

function withdraw(
    uint256 assets,
    address receiver,
    address owner
) public returns (uint256 shares) {
    shares = previewWithdraw(assets); // No need to check for rounding error, previewWithdraw rounds up.
    if (msg.sender != owner) {
        uint256 allowed = allowance[owner][msg.sender]; // Saves gas for limited approvals.
        if (allowed != type(uint256).max) allowance[owner][msg.sender] = allowed - shares;
    }
    _burn(owner, shares);
    emit Withdraw(msg.sender, receiver, owner, assets, shares);
    address(_asset).safeTransfer(receiver, assets);
}

function redeem(
    uint256 shares,
    address receiver,
    address owner
) public returns (uint256 assets) {
    if (msg.sender != owner) {
        uint256 allowed = allowance[owner][msg.sender]; // Saves gas for limited approvals.
        if (allowed != type(uint256).max) allowance[owner][msg.sender] = allowed - shares;
    }
    // Check for rounding error since we round down in previewRedeem.
    require((assets = previewRedeem(shares)) != 0, "ZERO_ASSETS");
    _burn(owner, shares);
    emit Withdraw(msg.sender, receiver, owner, assets, shares);
    address(_asset).safeTransfer(receiver, assets);
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

We recommend using ReentrancyGuard through the nonReentrant modifier.

Remediation Plan:

SOLVED: The THORSwap team added the nonReentrant modifier to the deposit, mint, withdraw and redeem functions.

3.3 (HAL-03) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the Owners and TSAggregator contracts there are management functions marked as public, but they are never directly called within the contract itself or in any of its descendants:

- `setOwner(address owner, bool active)public virtual isOwner` (Owners.sol#19)
- `setFee(uint256 _fee, address _feeRecipient)public` (TSAggregator.sol#26)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by descendants, it is better to mark them as `external` to reduce gas costs.

Remediation Plan:

SOLVED: The THORSwap team marked the `setOwner` and `setFee` functions as `external`.

AUTOMATED TESTING

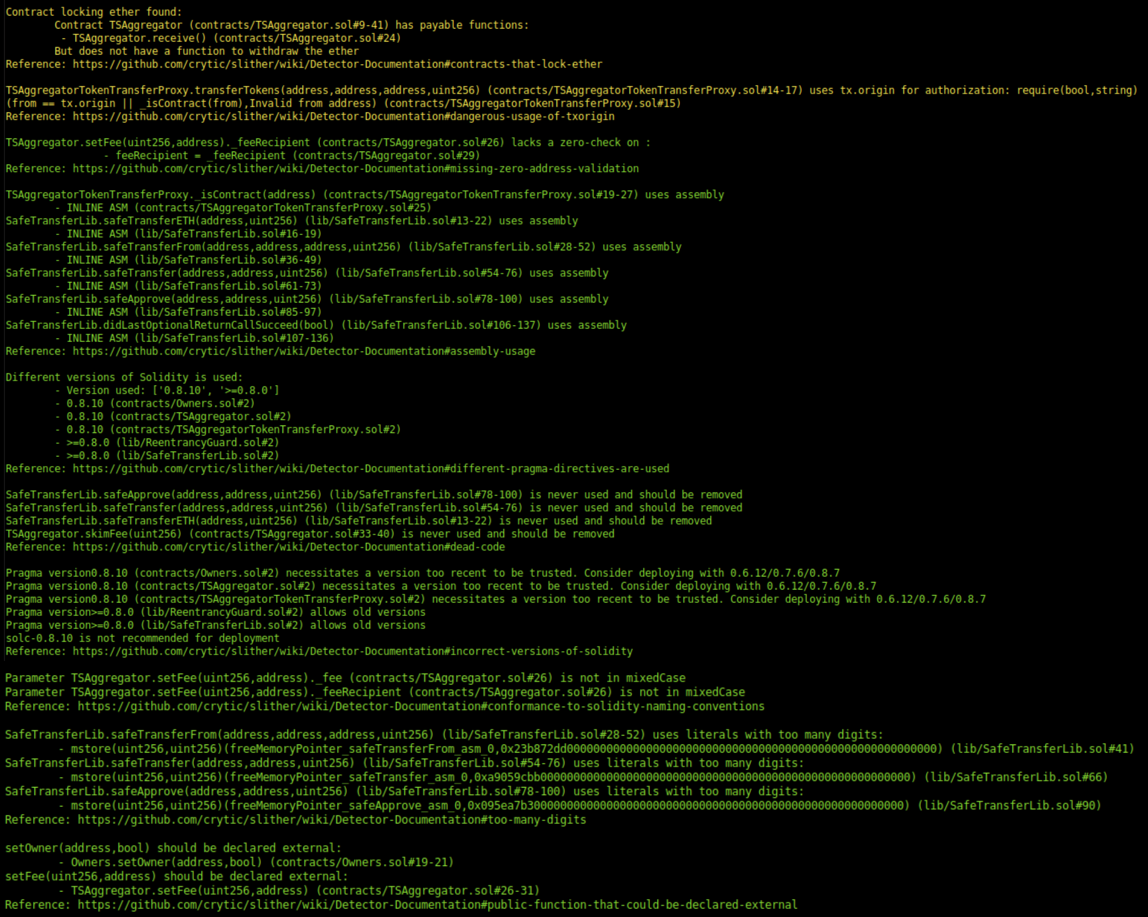
4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither Results:

TSAggregator.sol



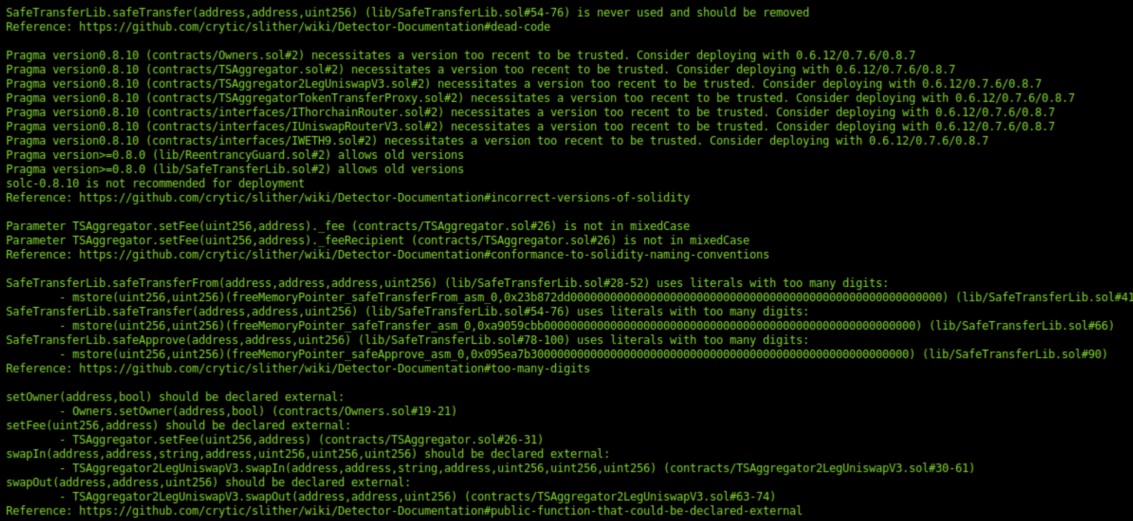
```
Contract locking ether found:
        Contract TSAggregator (contracts/TSAggregator.sol#9-41) has payable functions:
         - TSAggregator.receive() (contracts/TSAggregator.sol#24)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) is never used and should be removed
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) is never used and should be removed
TSAggregator.skimFee(uint256) (contracts/TSAggregator.sol#33-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TSAggregator2LegUniswapV2.sol

```
TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
TSAggregator2LegUniswapV2.constructor(address,address,address,address)._weth (contracts/TSAggregator2LegUniswapV2.sol#17) lacks a zero-check on :
                - weth = _weth (contracts/TSAggregator2LegUniswapV2.sol#19)
TSAggregator2LegUniswapV2.constructor(address,address,address,address)._legToken (contracts/TSAggregator2LegUniswapV2.sol#17) lacks a zero-check on :
                - legToken = _legToken (contracts/TSAggregator2LegUniswapV2.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregator2LegUniswapV2.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - 0.8.10 (contracts/interfaces/IThorchainRouter.sol#2)
        - 0.8.10 (contracts/interfaces/IUniswapRouterV2.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator2LegUniswapV2.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IThorchainRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IUniswapRouterV2.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
swapIn(address,address,string,address,uint256,uint256,uint256) should be declared external:
        - TSAggregator2LegUniswapV2.swapIn(address,address,string,address,uint256,uint256,uint256) (contracts/TSAggregator2LegUniswapV2.sol#24-57)
swapOut(address,address,uint256) should be declared external:
        - TSAggregator2LegUniswapV2.swapOut(address,address,uint256) (contracts/TSAggregator2LegUniswapV2.sol#59-71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

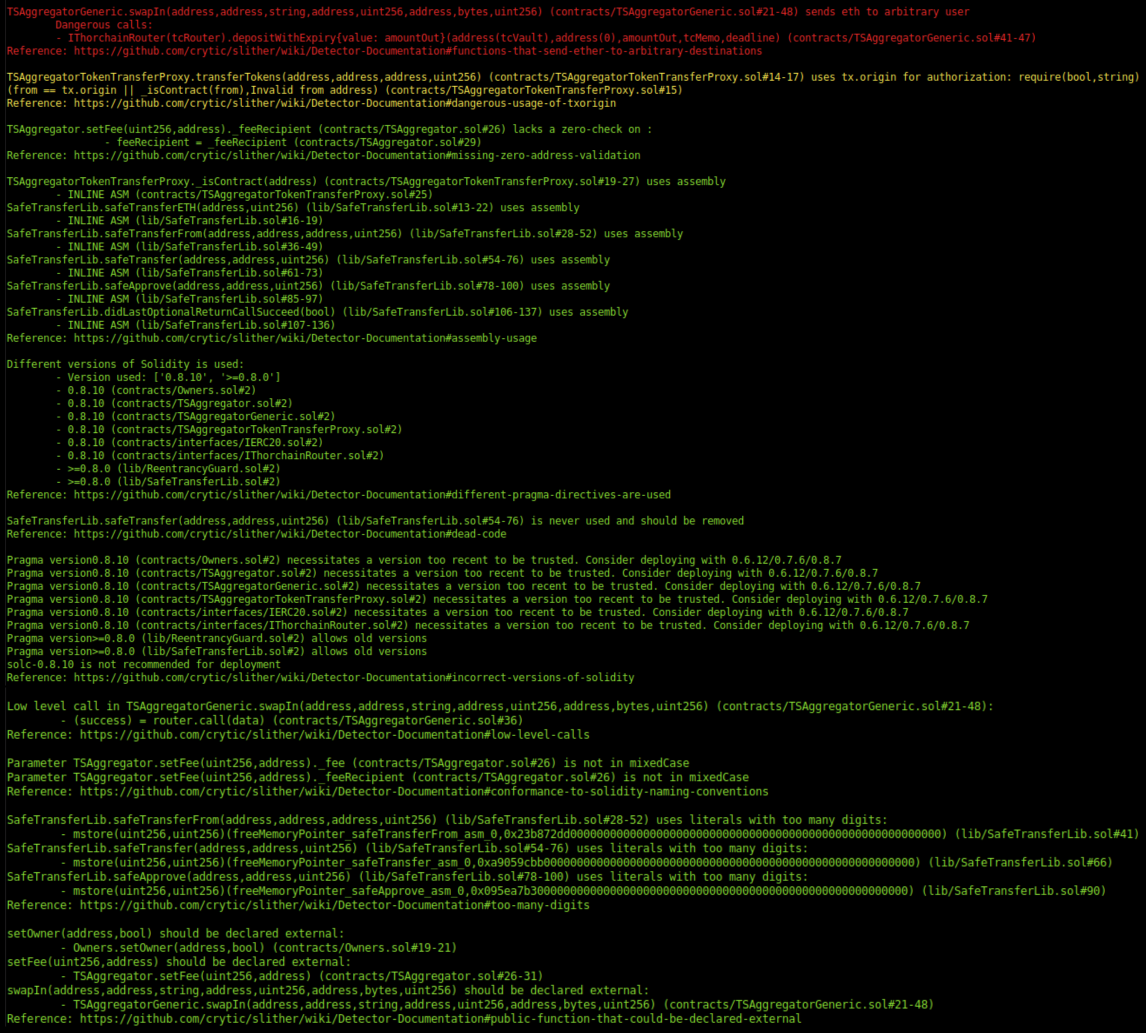
TSAggregator2LegUniswapV3.sol

```
TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregator2LegUniswapV3.swapOut(address,address,uint256) (contracts/TSAggregator2LegUniswapV3.sol#63-74) ignores return value by swapRouter.exactInput(IUniswapRouterV3.ExactInputParams
(path,to,type()(uint256).max,amount,amountOutMin)) (contracts/TSAggregator2LegUniswapV3.sol#67-73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
TSAggregator2LegUniswapV3.constructor(address,uint24,address,address,address,uint24)._legToken (contracts/TSAggregator2LegUniswapV3.sol#21) lacks a zero-check on :
                - legToken = _legToken (contracts/TSAggregator2LegUniswapV3.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregator2LegUniswapV3.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - 0.8.10 (contracts/interfaces/IThorchainRouter.sol#2)
        - 0.8.10 (contracts/interfaces/IUniswapRouterV3.sol#2)
        - 0.8.10 (contracts/interfaces/IWETH9.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```



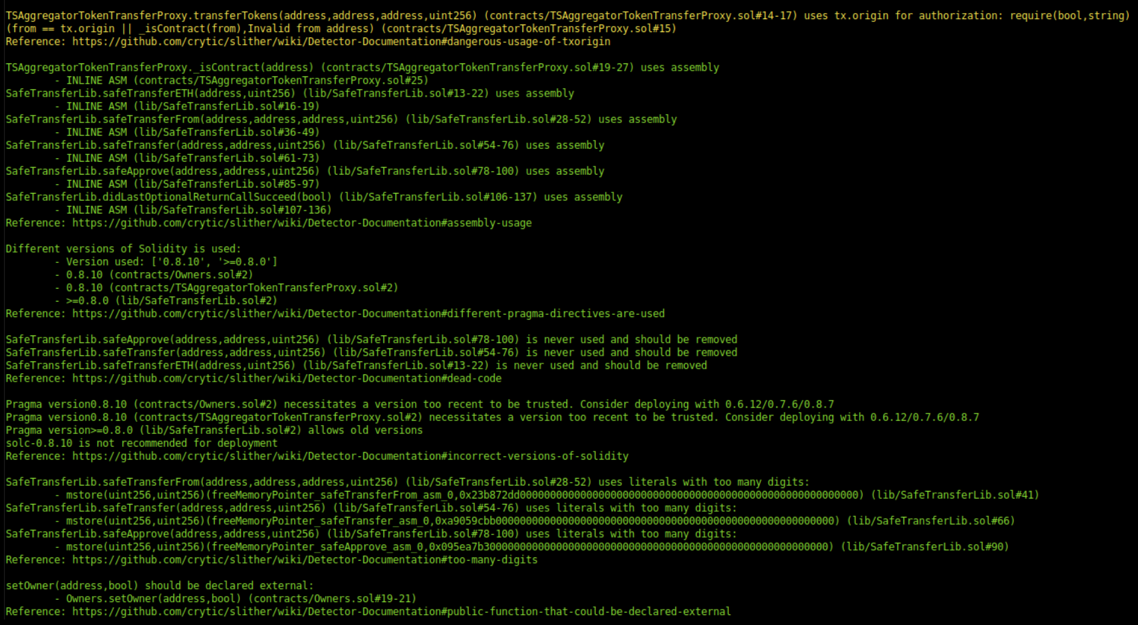
```
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator2LegUniswapV3.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IThorchainRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IUniswapRouterV3.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IWETH9.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
swapIn(address,address,string,address,uint256,uint256,uint256) should be declared external:
        - TSAggregator2LegUniswapV3.swapIn(address,address,string,address,uint256,uint256,uint256) (contracts/TSAggregator2LegUniswapV3.sol#30-61)
swapOut(address,address,uint256) should be declared external:
        - TSAggregator2LegUniswapV3.swapOut(address,address,uint256) (contracts/TSAggregator2LegUniswapV3.sol#63-74)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TSAggregatorGeneric.sol



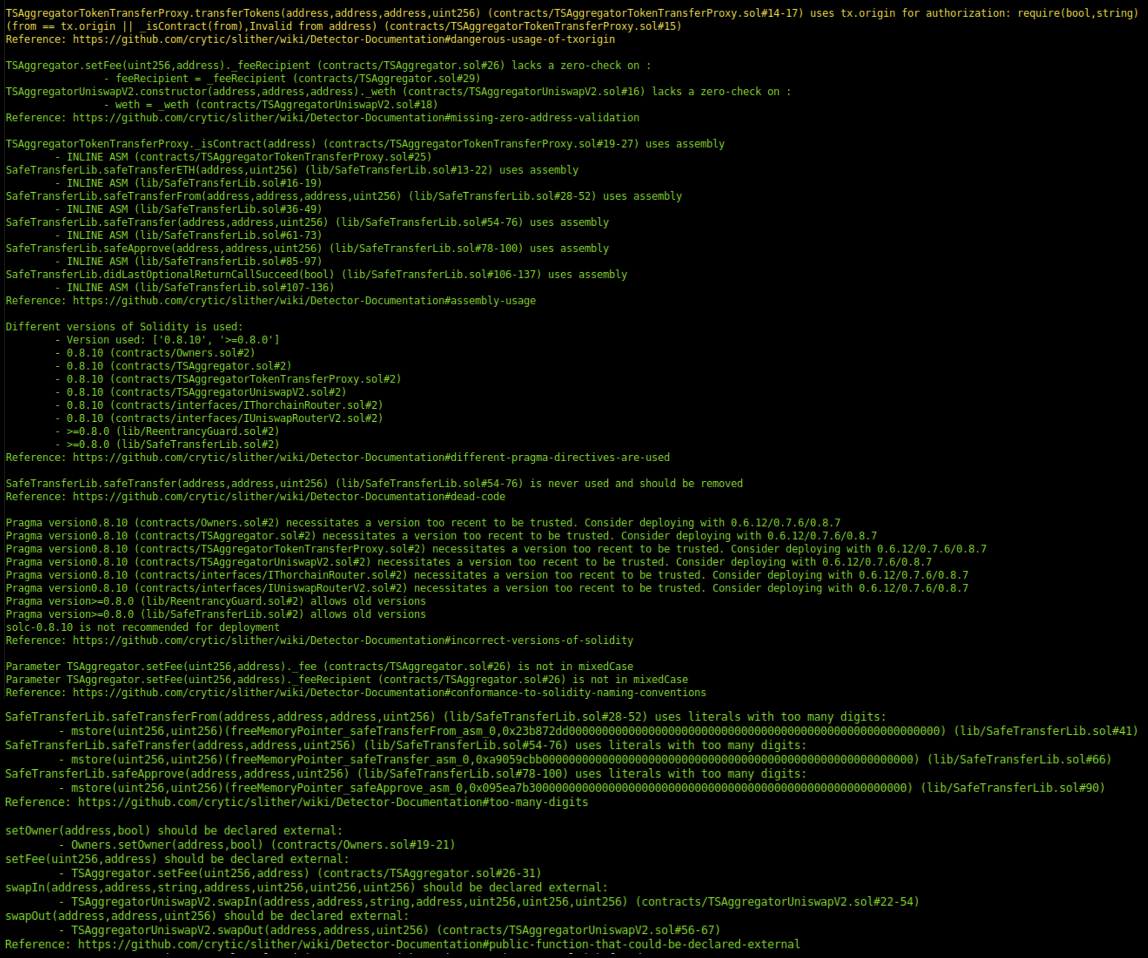
```
TSAggregatorGeneric.swapIn(address,address,string,address,uint256,address,bytes,uint256) (contracts/TSAggregatorGeneric.sol#21-48) sends eth to arbitrary user
        Dangerous calls:
        - IThorchainRouter(tcRouter).depositWithExpiry{value: amountOut}(address(tcVault),address(0),amountOut,tcMemo,deadline) (contracts/TSAggregatorGeneric.sol#41-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregatorGeneric.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - 0.8.10 (contracts/interfaces/IERC20.sol#2)
        - 0.8.10 (contracts/interfaces/IThorchainRouter.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorGeneric.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IThorchainRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in TSAggregatorGeneric.swapIn(address,address,string,address,uint256,address,bytes,uint256) (contracts/TSAggregatorGeneric.sol#21-48):
        - (success) = router.call(data) (contracts/TSAggregatorGeneric.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
swapIn(address,address,string,address,uint256,address,bytes,uint256) should be declared external:
        - TSAggregatorGeneric.swapIn(address,address,string,address,uint256,address,bytes,uint256) (contracts/TSAggregatorGeneric.sol#21-48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TSAggregatorTokenTransferProxy.sol



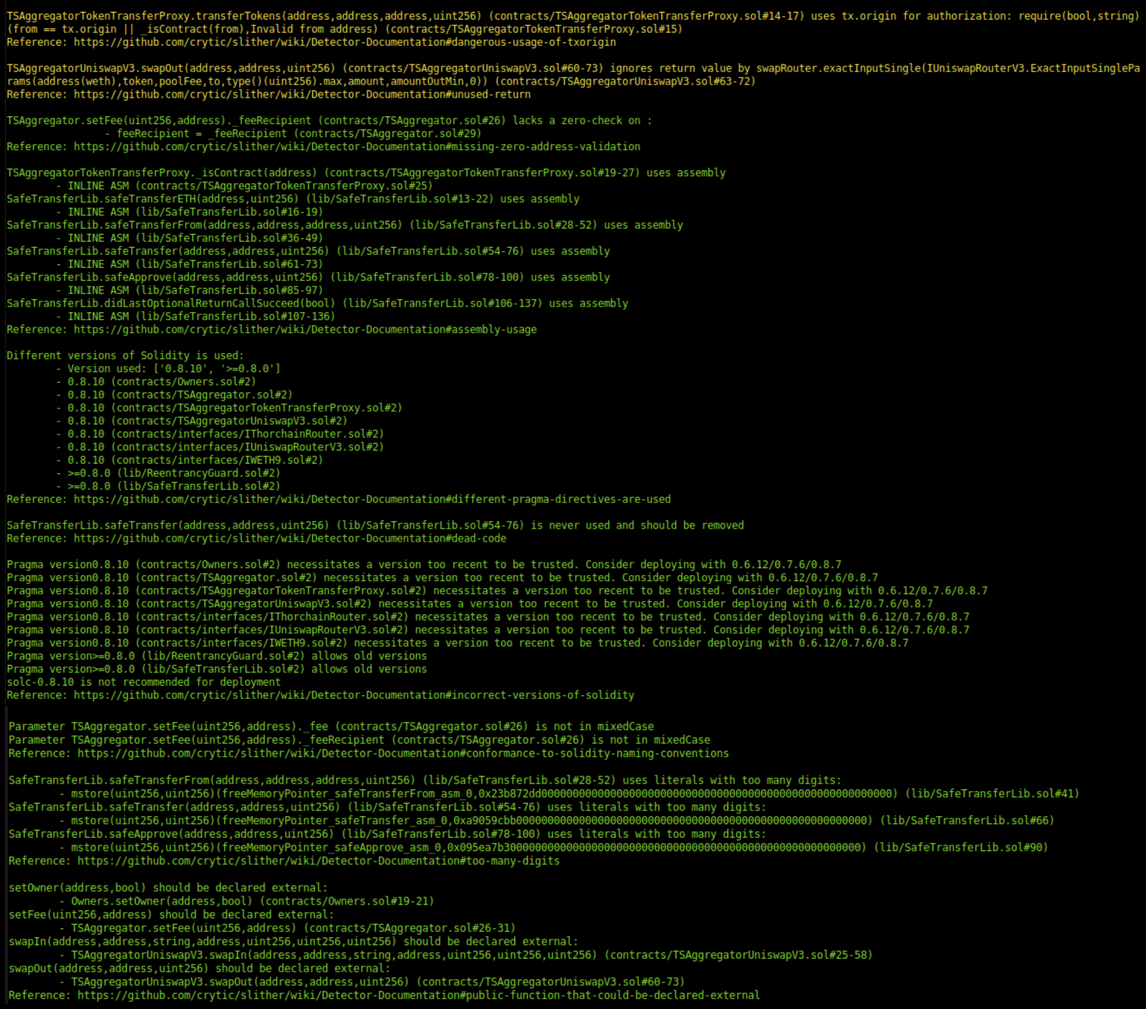
```
TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) is never used and should be removed
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TSAggregatorUniswapV2.sol



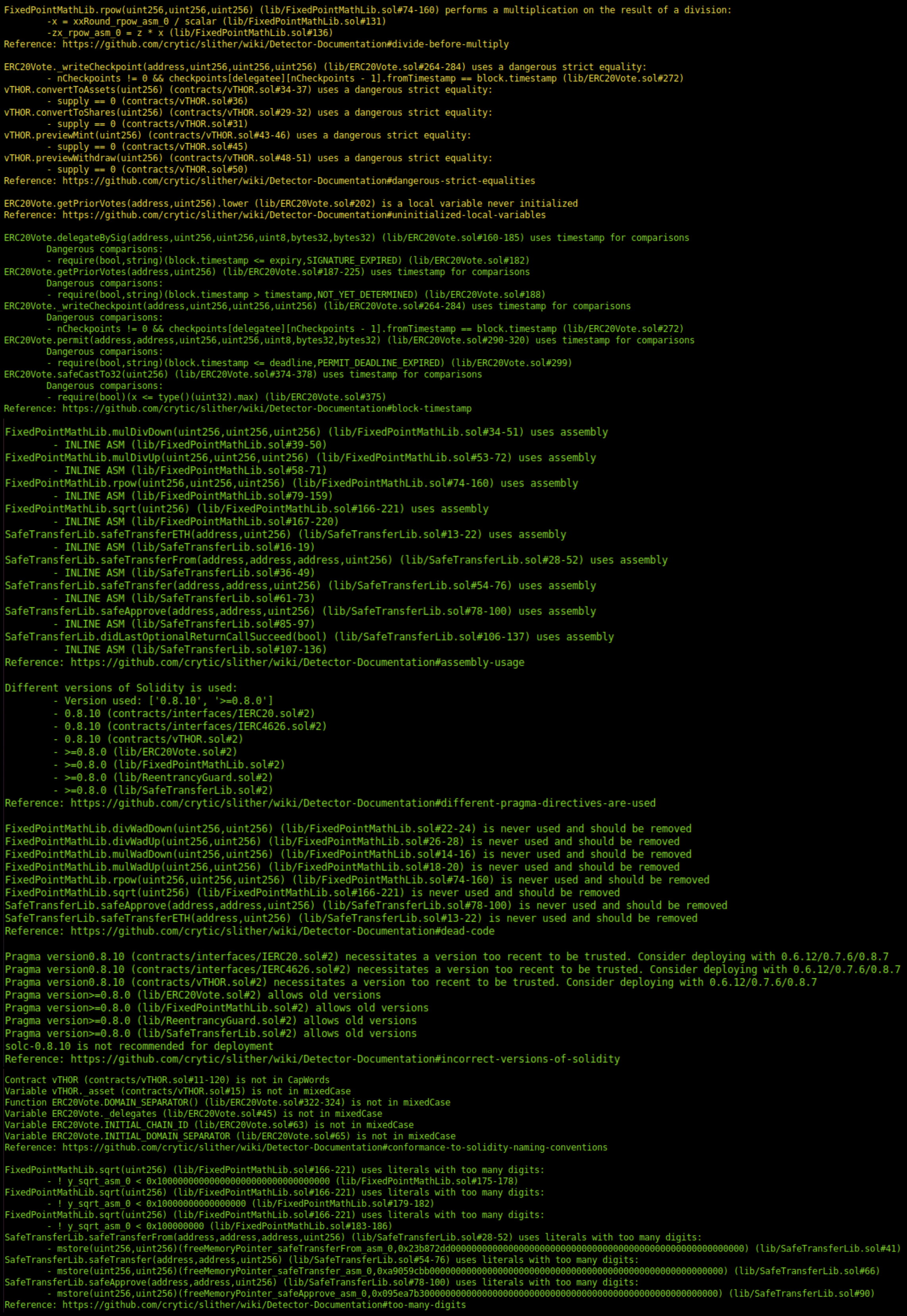
```
TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
TSAggregatorUniswapV2.constructor(address,address,address)._weth (contracts/TSAggregatorUniswapV2.sol#16) lacks a zero-check on :
                - weth = _weth (contracts/TSAggregatorUniswapV2.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - 0.8.10 (contracts/TSAggregatorUniswapV2.sol#2)
        - 0.8.10 (contracts/interfaces/IThorchainRouter.sol#2)
        - 0.8.10 (contracts/interfaces/IUniswapRouterV2.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorUniswapV2.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IThorchainRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IUniswapRouterV2.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
swapIn(address,address,string,address,uint256,uint256,uint256) should be declared external:
        - TSAggregatorUniswapV2.swapIn(address,address,string,address,uint256,uint256,uint256) (contracts/TSAggregatorUniswapV2.sol#22-54)
swapOut(address,address,uint256) should be declared external:
        - TSAggregatorUniswapV2.swapOut(address,address,uint256) (contracts/TSAggregatorUniswapV2.sol#56-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TSAggregatorUniswapV3.sol

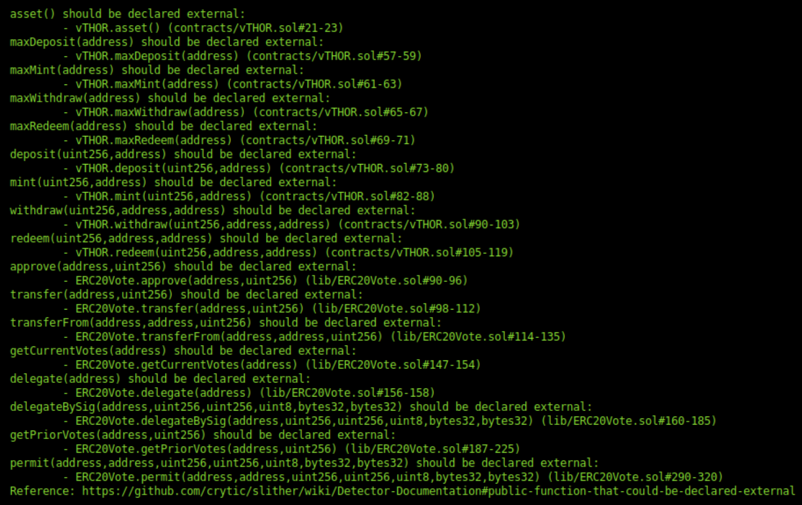


```
TSAggregatorTokenTransferProxy.transferTokens(address,address,address,uint256) (contracts/TSAggregatorTokenTransferProxy.sol#14-17) uses tx.origin for authorization: require(bool,string)
(from == tx.origin || _isContract(from),Invalid from address) (contracts/TSAggregatorTokenTransferProxy.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

TSAggregatorUniswapV3.swapOut(address,address,uint256) (contracts/TSAggregatorUniswapV3.sol#60-73) ignores return value by swapRouter.exactInputSingle(IUniswapRouterV3.ExactInputSinglePa
rams(address(weth),token,poolFee,to,type()(uint256).max,amount,amountOutMin,0)) (contracts/TSAggregatorUniswapV3.sol#63-72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) lacks a zero-check on :
                - feeRecipient = _feeRecipient (contracts/TSAggregator.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TSAggregatorTokenTransferProxy._isContract(address) (contracts/TSAggregatorTokenTransferProxy.sol#19-27) uses assembly
        - INLINE ASM (contracts/TSAggregatorTokenTransferProxy.sol#25)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/Owners.sol#2)
        - 0.8.10 (contracts/TSAggregator.sol#2)
        - 0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2)
        - 0.8.10 (contracts/TSAggregatorUniswapV3.sol#2)
        - 0.8.10 (contracts/interfaces/IThorchainRouter.sol#2)
        - 0.8.10 (contracts/interfaces/IUniswapRouterV3.sol#2)
        - 0.8.10 (contracts/interfaces/IWETH9.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/Owners.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorTokenTransferProxy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/TSAggregatorUniswapV3.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IThorchainRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IUniswapRouterV3.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IWETH9.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter TSAggregator.setFee(uint256,address)._fee (contracts/TSAggregator.sol#26) is not in mixedCase
Parameter TSAggregator.setFee(uint256,address)._feeRecipient (contracts/TSAggregator.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

setOwner(address,bool) should be declared external:
        - Owners.setOwner(address,bool) (contracts/Owners.sol#19-21)
setFee(uint256,address) should be declared external:
        - TSAggregator.setFee(uint256,address) (contracts/TSAggregator.sol#26-31)
swapIn(address,address,string,address,uint256,uint256,uint256) should be declared external:
        - TSAggregatorUniswapV3.swapIn(address,address,string,address,uint256,uint256,uint256) (contracts/TSAggregatorUniswapV3.sol#25-58)
swapOut(address,address,uint256) should be declared external:
        - TSAggregatorUniswapV3.swapOut(address,address,uint256) (contracts/TSAggregatorUniswapV3.sol#60-73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

vTHOR.sol



```
FixedPointMathLib.rpow(uint256,uint256,uint256) (lib/FixedPointMathLib.sol#74-160) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / scalar (lib/FixedPointMathLib.sol#131)
        -zx_rpow_asm_0 = z * x (lib/FixedPointMathLib.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

ERC20Vote._writeCheckpoint(address,uint256,uint256,uint256) (lib/ERC20Vote.sol#264-284) uses a dangerous strict equality:
        - nCheckpoints != 0 && checkpoints[delegatee][nCheckpoints - 1].fromTimestamp == block.timestamp (lib/ERC20Vote.sol#272)
vTHOR.convertToAssets(uint256) (contracts/vTHOR.sol#34-37) uses a dangerous strict equality:
        - supply == 0 (contracts/vTHOR.sol#36)
vTHOR.convertToShares(uint256) (contracts/vTHOR.sol#29-32) uses a dangerous strict equality:
        - supply == 0 (contracts/vTHOR.sol#31)
vTHOR.previewMint(uint256) (contracts/vTHOR.sol#43-46) uses a dangerous strict equality:
        - supply == 0 (contracts/vTHOR.sol#45)
vTHOR.previewWithdraw(uint256) (contracts/vTHOR.sol#48-51) uses a dangerous strict equality:
        - supply == 0 (contracts/vTHOR.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

ERC20Vote.getPriorVotes(address,uint256).lower (lib/ERC20Vote.sol#202) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC20Vote.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (lib/ERC20Vote.sol#160-185) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,SIGNATURE_EXPIRED) (lib/ERC20Vote.sol#182)
ERC20Vote.getPriorVotes(address,uint256) (lib/ERC20Vote.sol#187-225) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > timestamp,NOT_YET_DETERMINED) (lib/ERC20Vote.sol#188)
ERC20Vote._writeCheckpoint(address,uint256,uint256,uint256) (lib/ERC20Vote.sol#264-284) uses timestamp for comparisons
        Dangerous comparisons:
        - nCheckpoints != 0 && checkpoints[delegatee][nCheckpoints - 1].fromTimestamp == block.timestamp (lib/ERC20Vote.sol#272)
ERC20Vote.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (lib/ERC20Vote.sol#290-320) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,PERMIT_DEADLINE_EXPIRED) (lib/ERC20Vote.sol#299)
ERC20Vote.safeCastTo32(uint256) (lib/ERC20Vote.sol#374-378) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(x <= type()(uint32).max) (lib/ERC20Vote.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

FixedPointMathLib.mulDivDown(uint256,uint256,uint256) (lib/FixedPointMathLib.sol#34-51) uses assembly
        - INLINE ASM (lib/FixedPointMathLib.sol#39-50)
FixedPointMathLib.mulDivUp(uint256,uint256,uint256) (lib/FixedPointMathLib.sol#53-72) uses assembly
        - INLINE ASM (lib/FixedPointMathLib.sol#58-71)
FixedPointMathLib.rpow(uint256,uint256,uint256) (lib/FixedPointMathLib.sol#74-160) uses assembly
        - INLINE ASM (lib/FixedPointMathLib.sol#79-159)
FixedPointMathLib.sqrt(uint256) (lib/FixedPointMathLib.sol#166-221) uses assembly
        - INLINE ASM (lib/FixedPointMathLib.sol#167-220)
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#16-19)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#36-49)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#61-73)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#85-97)
SafeTransferLib.didLastOptionalReturnCallSucceed(bool) (lib/SafeTransferLib.sol#106-137) uses assembly
        - INLINE ASM (lib/SafeTransferLib.sol#107-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.10', '>=0.8.0']
        - 0.8.10 (contracts/interfaces/IERC20.sol#2)
        - 0.8.10 (contracts/interfaces/IERC4626.sol#2)
        - 0.8.10 (contracts/vTHOR.sol#2)
        - >=0.8.0 (lib/ERC20Vote.sol#2)
        - >=0.8.0 (lib/FixedPointMathLib.sol#2)
        - >=0.8.0 (lib/ReentrancyGuard.sol#2)
        - >=0.8.0 (lib/SafeTransferLib.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

FixedPointMathLib.divWadDown(uint256,uint256) (lib/FixedPointMathLib.sol#22-24) is never used and should be removed
FixedPointMathLib.divWadUp(uint256,uint256) (lib/FixedPointMathLib.sol#26-28) is never used and should be removed
FixedPointMathLib.mulWadDown(uint256,uint256) (lib/FixedPointMathLib.sol#14-16) is never used and should be removed
FixedPointMathLib.mulWadUp(uint256,uint256) (lib/FixedPointMathLib.sol#18-20) is never used and should be removed
FixedPointMathLib.rpow(uint256,uint256,uint256) (lib/FixedPointMathLib.sol#74-160) is never used and should be removed
FixedPointMathLib.sqrt(uint256) (lib/FixedPointMathLib.sol#166-221) is never used and should be removed
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (lib/SafeTransferLib.sol#13-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.10 (contracts/interfaces/IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/vTHOR.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.8.0 (lib/ERC20Vote.sol#2) allows old versions
Pragma version>=0.8.0 (lib/FixedPointMathLib.sol#2) allows old versions
Pragma version>=0.8.0 (lib/ReentrancyGuard.sol#2) allows old versions
Pragma version>=0.8.0 (lib/SafeTransferLib.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract vTHOR (contracts/vTHOR.sol#11-120) is not in CapWords
Variable vTHOR._asset (contracts/vTHOR.sol#15) is not in mixedCase
Function ERC20Vote.DOMAIN_SEPARATOR() (lib/ERC20Vote.sol#322-324) is not in mixedCase
Variable ERC20Vote._delegates (lib/ERC20Vote.sol#45) is not in mixedCase
Variable ERC20Vote.INITIAL_CHAIN_ID (lib/ERC20Vote.sol#63) is not in mixedCase
Variable ERC20Vote.INITIAL_DOMAIN_SEPARATOR (lib/ERC20Vote.sol#65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

FixedPointMathLib.sqrt(uint256) (lib/FixedPointMathLib.sol#166-221) uses literals with too many digits:
        - ! y_sqrt_asm_0 < 0x100000000000000000000000000000000 (lib/FixedPointMathLib.sol#175-178)
FixedPointMathLib.sqrt(uint256) (lib/FixedPointMathLib.sol#166-221) uses literals with too many digits:
        - ! y_sqrt_asm_0 < 0x10000000000000000 (lib/FixedPointMathLib.sol#179-182)
FixedPointMathLib.sqrt(uint256) (lib/FixedPointMathLib.sol#166-221) uses literals with too many digits:
        - ! y_sqrt_asm_0 < 0x100000000 (lib/FixedPointMathLib.sol#183-186)
SafeTransferLib.safeTransferFrom(address,address,address,uint256) (lib/SafeTransferLib.sol#28-52) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#41)
SafeTransferLib.safeTransfer(address,address,uint256) (lib/SafeTransferLib.sol#54-76) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#66)
SafeTransferLib.safeApprove(address,address,uint256) (lib/SafeTransferLib.sol#78-100) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (lib/SafeTransferLib.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```



```
asset() should be declared external:
	- vTHOR.asset() (contracts/vTHOR.sol#21-23)
maxDeposit(address) should be declared external:
	- vTHOR.maxDeposit(address) (contracts/vTHOR.sol#57-59)
maxMint(address) should be declared external:
	- vTHOR.maxMint(address) (contracts/vTHOR.sol#61-63)
maxWithdraw(address) should be declared external:
	- vTHOR.maxWithdraw(address) (contracts/vTHOR.sol#65-67)
maxRedeem(address) should be declared external:
	- vTHOR.maxRedeem(address) (contracts/vTHOR.sol#69-71)
deposit(uint256,address) should be declared external:
	- vTHOR.deposit(uint256,address) (contracts/vTHOR.sol#73-80)
mint(uint256,address) should be declared external:
	- vTHOR.mint(uint256,address) (contracts/vTHOR.sol#82-88)
withdraw(uint256,address,address) should be declared external:
	- vTHOR.withdraw(uint256,address,address) (contracts/vTHOR.sol#90-103)
redeem(uint256,address,address) should be declared external:
	- vTHOR.redeem(uint256,address,address) (contracts/vTHOR.sol#105-119)
approve(address,uint256) should be declared external:
	- ERC20Vote.approve(address,uint256) (lib/ERC20Vote.sol#90-96)
transfer(address,uint256) should be declared external:
	- ERC20Vote.transfer(address,uint256) (lib/ERC20Vote.sol#98-112)
transferFrom(address,address,uint256) should be declared external:
	- ERC20Vote.transferFrom(address,address,uint256) (lib/ERC20Vote.sol#114-135)
getCurrentVotes(address) should be declared external:
	- ERC20Vote.getCurrentVotes(address) (lib/ERC20Vote.sol#147-154)
delegate(address) should be declared external:
	- ERC20Vote.delegate(address) (lib/ERC20Vote.sol#156-158)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
	- ERC20Vote.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (lib/ERC20Vote.sol#160-185)
getPriorVotes(address,uint256) should be declared external:
	- ERC20Vote.getPriorVotes(address,uint256) (lib/ERC20Vote.sol#187-225)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
	- ERC20Vote.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (lib/ERC20Vote.sol#290-320)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.
- The authorization (use of tx.origin) and “send eth to arbitrary user” issues are all false positives.

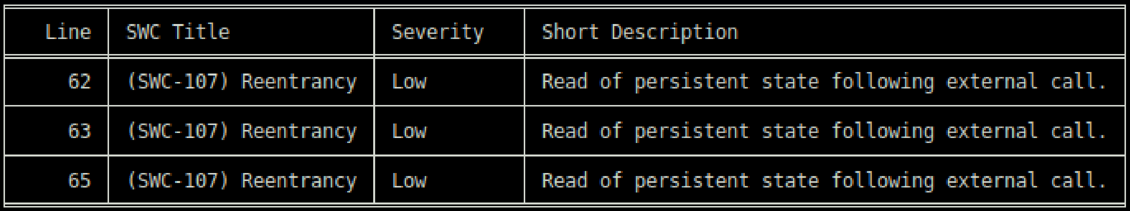
4.2 AUTOMATED SECURITY SCAN

MYTHX:

Halborn used automated security scanners to assist with detecting well-known security issues and to identify low-hanging fruits on the targets for this engagement. MythX, a security analysis service for Ethereum smart contracts, is among the tools used. MythX was used to scan all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

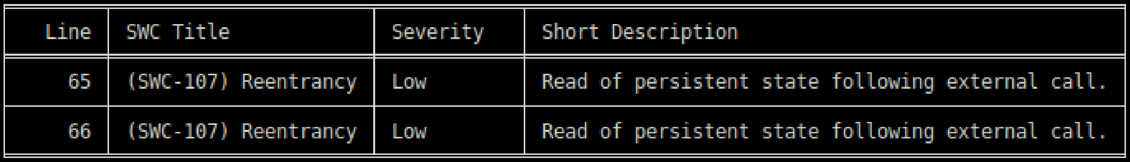
Results:

TSAggregator2LegUniswapV2.sol



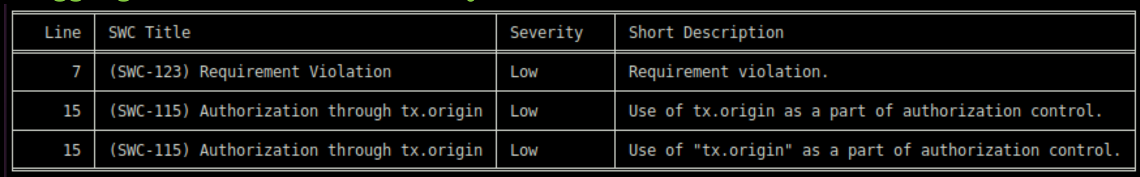
Line	SWC Title	Severity	Short Description
62	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
63	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
65	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

TSAggregator2LegUniswapV3.sol



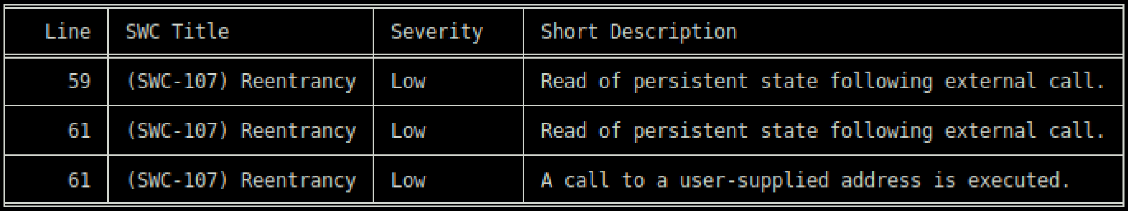
Line	SWC Title	Severity	Short Description
65	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
66	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

TSAggregatorTokenTransferProxy.sol



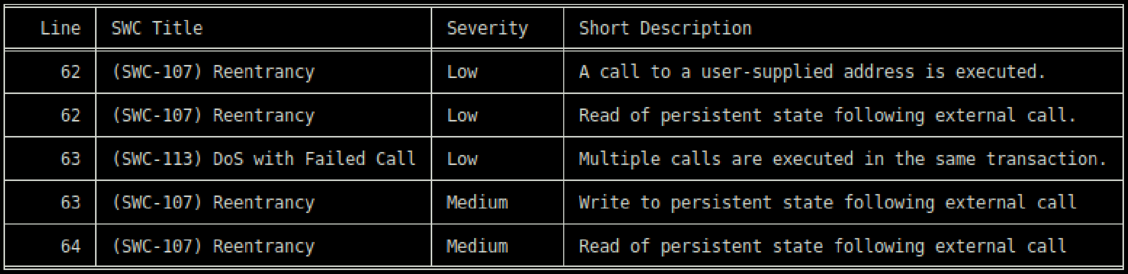
Line	SWC Title	Severity	Short Description
7	(SWC-123) Requirement Violation	Low	Requirement violation.
15	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.
15	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.

TSAggregatorUniswapV2.sol



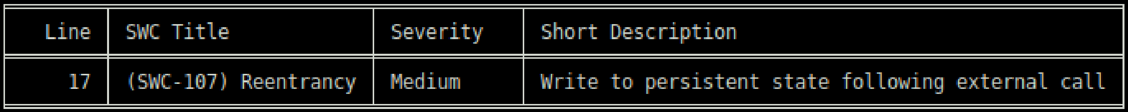
Line	SWC Title	Severity	Short Description
59	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
61	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
61	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.

TSAggregatorUniswapV3.sol



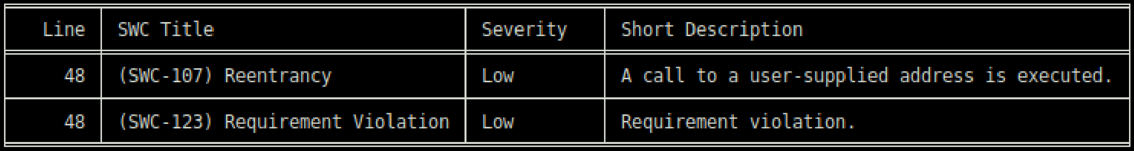
Line	SWC Title	Severity	Short Description
62	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.
62	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
63	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
63	(SWC-107) Reentrancy	Medium	Write to persistent state following external call
64	(SWC-107) Reentrancy	Medium	Read of persistent state following external call

ReentrancyGuard.sol



Line	SWC Title	Severity	Short Description
17	(SWC-107) Reentrancy	Medium	Write to persistent state following external call

SafeTransferLib.sol



Line	SWC Title	Severity	Short Description
48	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.
48	(SWC-123) Requirement Violation	Low	Requirement violation.

- No major issues found by MythX.
- The reentrancy, authorization, DOS and requirement violation issues are all false positives.

THANK YOU FOR CHOOSING

HALBORN