NEBULA

FATBULL

Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Report

December 2021

Disclaimer

Report Summary

Title	**FATBULL Smart Contract Audit**		
Project Owner	FATBULL		
Type	Public		
Reviewed by	Vatsal Raychura	**Revision date**	14/12/2021
Approved by	eNebula Solutions Private Limited	**Approval date**	14/12/2021
		Nº Pages	**45**

Overview

Background

FATBULL's team requested that eNebula Solutions perform an Extensive Smart Contract audit of their Fat Smart Contract.

Project Dates

The following is the project schedule for this review and report:

- **December 14**: Smart Contract Review Completed *(Completed)*
- **December 14**: Delivery of Smart Contract Audit Report *(Completed)*

Review Team

The following eNebula Solutions team member participated in this review:

- Sejal Barad, Security Researcher and Engineer
- Vatsal Raychura, Security Researcher and Engineer

Coverage

Target Specification and Revision

For this audit, we performed research, investigation, and review of the smart contract of FATBULL.

The following documentation repositories were considered in-scope for the review:

- FATBULL Project:

FAT.txt

Introduction

Given the opportunity to review FATBULL Project's smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high issues found related to business logic, security or performance.

About FATBULL: -

Item	Description
Issuer	FATBULL
Website	www.fatbulltoken.io
Platform	Solidity
Audit Method	Whitebox
Latest Audit Report	December 14, 2021

The Test Method Information: -

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

The Full List of Check Items:

Category	Check Item
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	MONEY-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead of Transfer
	Costly Loop
	(Unsafe) Use of Untrusted Libraries
	(Unsafe) Use of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
	Business Logics Review

Advanced DeFi Scrutiny	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in This Audit:

Category	Summary
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.
Resource Management	Weaknesses in this category are related to improper management of system resources.

Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

Findings

Summary

Here is a summary of our findings after analyzing the FATBULL's Smart Contract. During the first phase of our audit, we studied the smart contract sourcecode and ran our in-house static code analyzer through the Specific tool. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tool. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	**0**
High	**1**
Medium	**0**
Low	**5**
Total	**7(1 Compilation Error)**

We have so far identified that there are potential issues with severity of **0 Critical, 1 High, 0 Medium, and 5 Low**. Overall, these smart contracts are well- designed and engineered. And for the issues the recommendations to resolve them have been mentioned in the detailed findings section.

Functional Overview

($) = payable function # = non-constant function	[Pub] public [Ext] external [Prv] private [Int] internal

```
+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

 +  Context
   - [Int] _msgSender
   - [Int] _msgData

 + [Int] IUniswapV2Router01
   - [Ext] factory
   - [Ext] WETH
   - [Ext] addLiquidity #
   - [Ext] addLiquidityETH ($)
   - [Ext] removeLiquidity #
   - [Ext] removeLiquidityETH #
   - [Ext] removeLiquidityWithPermit #
   - [Ext] removeLiquidityETHWithPermit #
   - [Ext] swapExactTokensForTokens #
```

```
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens ($)
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] mint #
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Lib] IterableMapping
  - [Pub] get
  - [Pub] getIndexOfKey
  - [Pub] getKeyAtIndex
  - [Pub] size
  - [Pub] set #
  - [Pub] remove #

+  Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner

+ [Int] IDividendPayingTokenOptional
  - [Ext] withdrawableDividendOf
  - [Ext] withdrawnDividendOf
  - [Ext] accumulativeDividendOf

+ [Int] IDividendPayingToken
  - [Ext] dividendOf
  - [Ext] distributeDividends ($)
  - [Ext] withdrawDividend #

+ [Lib] SafeMathInt
  - [Int] mul
  - [Int] div
  - [Int] sub
  - [Int] add
  - [Int] toUint256Safe

+ [Lib] SafeMathUint
  - [Int] toInt256Safe
```

```
+  ERC20 (Context, IERC20)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #
  - [Int] _approve #
  - [Int] _setupDecimals #
  - [Int] _beforeTokenTransfer #

+ [Lib] SafeMath
  - [Int] tryAdd
  - [Int] trySub
  - [Int] tryMul
  - [Int] tryDiv
  - [Int] tryMod
  - [Int] add
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] mod
  - [Int] sub
  - [Int] div
  - [Int] mod

 + DividendPayingToken (ERC20, IDividendPayingToken,
IDividendPayingTokenOptional)
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Ext] <Fallback> ($)
  - [Pub] updateMasterContract #
    - modifiers: onlyMaster
  - [Pub] updateUniswapV2Router #
    - modifiers: onlyMaster
  - [Pub] updateDividendTokenUser #
    - modifiers: onlyMaster
  - [Pub] getDividendTokenUser
  - [Pub] distributeDividends ($)
    - modifiers: onlyMaster
  - [Pub] distributeTokenDividends #
    - modifiers: onlyMaster
  - [Pub] withdrawDividend #
  - [Prv] swapEthForTokens #
  - [Int] _withdrawDividendOfUser #
  - [Pub] dividendOf
  - [Pub] withdrawableDividendOf
  - [Pub] withdrawnDividendOf
  - [Pub] accumulativeDividendOf
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #
  - [Int] _setBalance #
```

```
+ FAT (ERC20, Ownable)
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Ext] <Fallback> ($)
  - [Int] restoreFees #
  - [Ext] swapAndLiquifyOwner #
    - modifiers: onlyOwner
  - [Ext] updateDividendTokenUser #
  - [Ext] getDividendTokenUser
  - [Ext] updatedividendTime #
    - modifiers: onlyOwner
  - [Ext] updateBuyBackMode #
    - modifiers: onlyOwner
  - [Ext] updateTradingEnabledTime #
    - modifiers: onlyOwner
  - [Ext] updateMinimumBalanceForDividends #
    - modifiers: onlyOwner
  - [Ext] updateMaxWalletAmount #
    - modifiers: onlyOwner
  - [Ext] updateSwapAtAmount #
    - modifiers: onlyOwner
  - [Ext] updateMarketAddress #
    - modifiers: onlyOwner
  - [Ext] updateCharityAddress #
    - modifiers: onlyOwner
  - [Ext] updateBuyBackAddress #
    - modifiers: onlyOwner
  - [Ext] updateMarketTokenFeeAddress #
    - modifiers: onlyOwner
  - [Ext] updateCharityTokenFeeAddress #
    - modifiers: onlyOwner
- [Ext] updateBuyBackTokenFeeAddress #
  - modifiers: onlyOwner
- [Ext] updateFees #
  - modifiers: onlyOwner
- [Ext] updateBuyFees #
  - modifiers: onlyOwner
- [Ext] updateSellFees #
  - modifiers: onlyOwner
- [Ext] updateDividendTracker #
  - modifiers: onlyOwner
- [Ext] updateUniswapV2Router #
  - modifiers: onlyOwner
- [Pub] excludeFromFees #
  - modifiers: onlyOwner
- [Pub] blacklistAddress #
  - modifiers: onlyOwner
- [Pub] excludeFromDividends #
  - modifiers: onlyOwner
- [Pub] enableDividends #
  - modifiers: onlyOwner
- [Ext] excludeMultipleAccountsFromFees #
  - modifiers: onlyOwner
- [Ext] setAutomatedMarketMakerPair #
  - modifiers: onlyOwner
- [Prv] _setAutomatedMarketMakerPair #
- [Pub] updateGasForProcessing #
  - modifiers: onlyOwner
- [Ext] updateClaimWait #
  - modifiers: onlyOwner
- [Ext] getClaimWait
- [Ext] getTotalDividendsDistributed
```

```
  - [Pub] isExcludedFromFees
  - [Pub] withdrawableDividendOf
  - [Pub] dividendTokenBalanceOf
  - [Ext] getAccountDividendsInfo
  - [Ext] getAccountDividendsInfoAtIndex
  - [Ext] processDividendTracker #
  - [Ext] claim #
  - [Ext] getLastProcessedIndex
  - [Ext] getNumberOfDividendTokenHolders
  - [Pub] getTradingIsEnabled
  - [Prv] swapAndLiquify #
  - [Prv] swapEthForTokens #
  - [Prv] swapTokensForEth #
  - [Prv] swapTokensForTokens #
  - [Int] addLiquidity #
  - [Prv] swapAndSendDividends #
  - [Int] _transfer #

+  FATDividendTracker (DividendPayingToken, Ownable)
  - [Pub] <Constructor> #
    - modifiers: DividendPayingToken
  - [Ext] updateMinimumBalanceForDividends #
    - modifiers: onlyOwner
  - [Ext] updateTokenForDividend #
    - modifiers: onlyOwner
  - [Int] _transfer #
  - [Pub] withdrawDividend #
  - [Ext] excludeFromDividends #
    - modifiers: onlyOwner
  - [Ext] enableDividends #
    - modifiers: onlyOwner
- [Ext] updateClaimWait #
   - modifiers: onlyOwner
- [Ext] getLastProcessedIndex
- [Ext] getNumberOfTokenHolders
- [Pub] getAccount
- [Pub] getAccountAtIndex
- [Prv] canAutoClaim
- [Ext] setBalance #
   - modifiers: onlyOwner
- [Pub] process #
- [Pub] processAccount #
   - modifiers: onlyOwner
```

Detailed Results

Compilations Issues.

1. DeclarationError: Undeclared identifier.

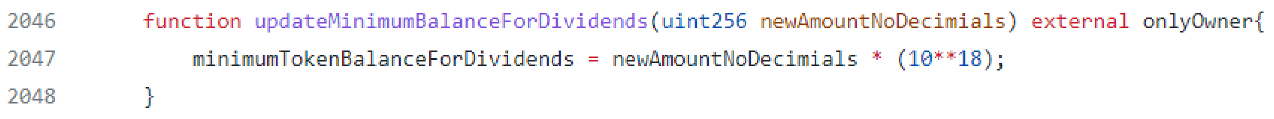
- Description: In the function transfer the require statement – “ require(amount <= maxSellTransactionAmount, "Sell transfer amount exceeds the maxSellTransactionAmount."); ” has undeclared identifier – ‘maxSellTransactionAmount’. Due to this the whole code cannot be compiled as having the DeclarationError.
- Recommendation: Please declare the undeclared identifier.

“Please Note that, during our audit we’ve assumed the declaration has been done properly and after that we’ve started the analysis and testing.”

Issues Checking Status

1. Integer Overflow and Underflow

- SWC ID:101
- Severity: High
- Location: Fat.sol
- Relationships: CWE-682: Incorrect Calculation
- Description: The arithmetic operation can overflow. It is possible to cause an arithmetic overflow. Prevent the overflow by constraining inputs using the require() statement or use the OpenZeppelin SafeMath library for integer arithmetic operations. Refer to the transaction trace generated for this issue to reproduce the overflow.

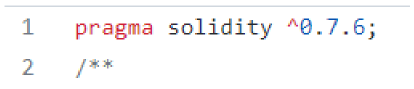


```
function updateMinimumBalanceForDividends(uint256 newAmountNoDecimials) external onlyOwner{
    minimumTokenBalanceForDividends = newAmountNoDecimials * (10**18);
}
```

- Remediations: It is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system.

2. Floating Pragma

- SWC ID:103
- Severity: Low
- Location: Fat.sol
- Relationships: CWE-664: Improper Control of a Resource Through its Lifetime
- Description: A floating pragma is set. The current pragma Solidity directive is ""^0.7.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.



```
pragma solidity ^0.7.6;
/**
```

- Remediations: Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

3. Reentrancy

- SWC ID:107
- Severity: Low(Impact)
- Location: Fat.sol
- Relationships: CWE-841: Improper Enforcement of Behavioral Workflow
- Description: Write to persistent state following external call. The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

```
    function owner() public view virtual returns (address) {
        return _owner;
    }
```

```
    function balanceOf(address account) public view virtual override returns (uint256) {
        return _balances[account];
    }
```

```
  function withdrawableDividendOf(address _owner) public view override returns(uint256) {
    return accumulativeDividendOf(_owner).sub(withdrawnDividends[_owner]);
  }
```

```
  function accumulativeDividendOf(address _owner) public view override returns(uint256) {
    return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
      .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
  }
```

```
        account = _account;

        index = tokenHoldersMap.getIndexOfKey(account);

        iterationsUntilProcessed = -1;

        if(index >= 0) {
            if(uint256(index) > lastProcessedIndex) {
                iterationsUntilProcessed = index.sub(int256(lastProcessedIndex));
            }
```

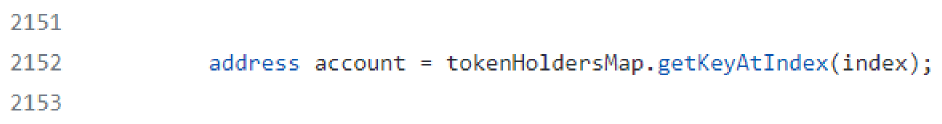
```

        lastClaimTime = lastClaimTimes[account];

```

```

            address account = tokenHoldersMap.getKeyAtIndex(index);

```

- Remediations: The best practices to avoid Reentrancy weaknesses are:
    - Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern
    - Use a reentrancy lock (ie. OpenZeppelin's ReentrancyGuard.

4. DoS with Failed Call

- SWC ID:113
- Severity: Low
- Location: Fat.sol
- Relationships: CWE-703: Improper Check or Handling of Exceptional Conditions
- Description: Multiple calls are executed in the same transaction. This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

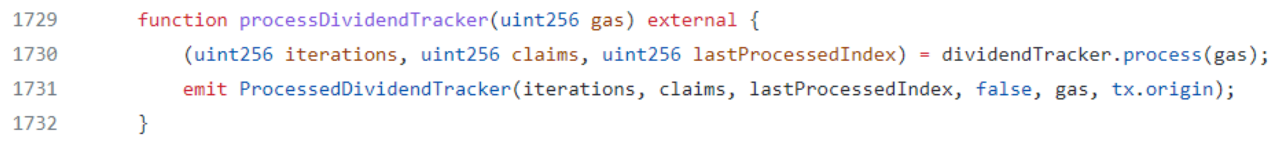


```
            address account = tokenHoldersMap.getKeyAtIndex(index);
```

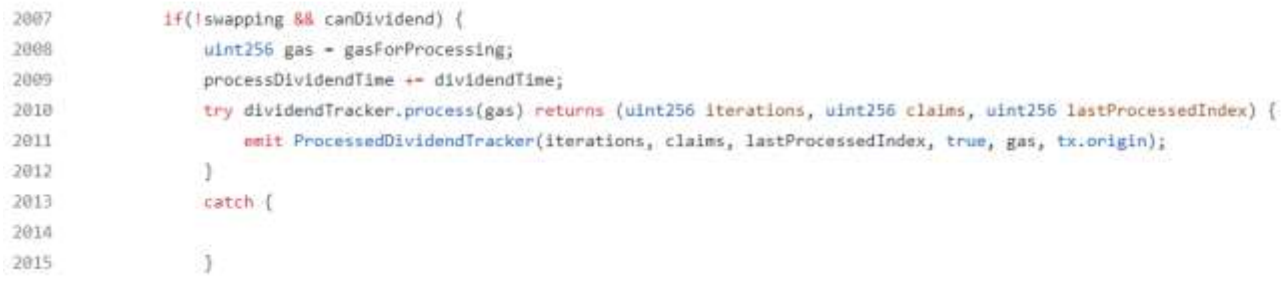
- Remediations: It is recommended to follow call best practices:
  - ➢ Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
  - ➢ Always assume that external calls can fail
  - ➢ Implement the contract logic to handle failed calls

5. Authorization through tx.origin

- SWC ID:115
- Severity: Low
- Location: Fat.sol
- Relationships: CWE-477: Use of Obsolete Function
- Description: Use of "tx.origin" as a part of authorization control. Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.



```
function processDividendTracker(uint256 gas) external {
    (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) = dividendTracker.process(gas);
    emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
}
```

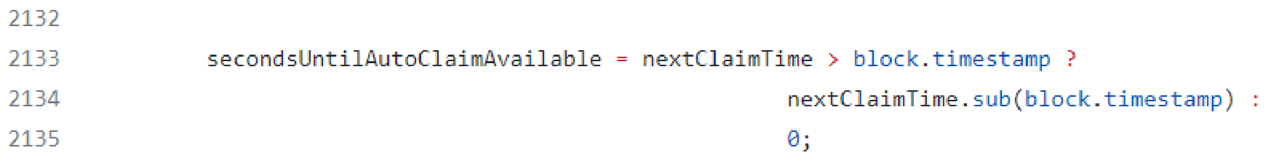


```
if(!swapping && canDividend) {
    uint256 gas = gasForProcessing;
    processDividendTime += dividendTime;
    try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) {
        emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas, tx.origin);
    }
    catch {

    }
```

- Remediations: tx.origin should not be used for authorization. Use msg.sender instead.

6. Block values as a proxy for time

- SWC ID:116
- Severity: Low
- Location: Fat.sol
- Relationships: CWE-829: Inclusion of Functionality from Untrusted Control Sphere
- Description: A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

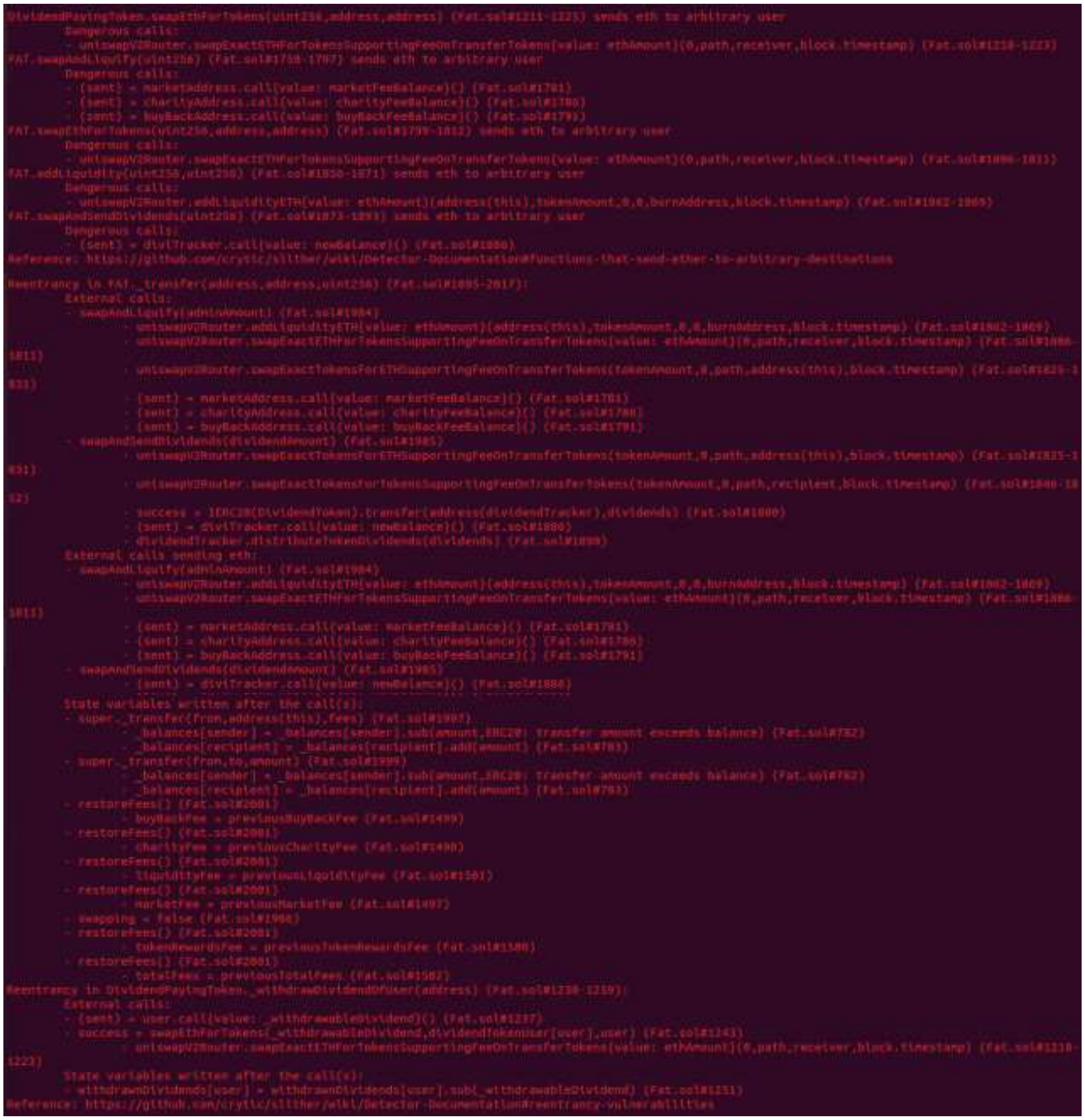


```

        secondsUntilAutoClaimAvailable = nextClaimTime > block.timestamp ?
                                                    nextClaimTime.sub(block.timestamp) :
                                                    0;
```

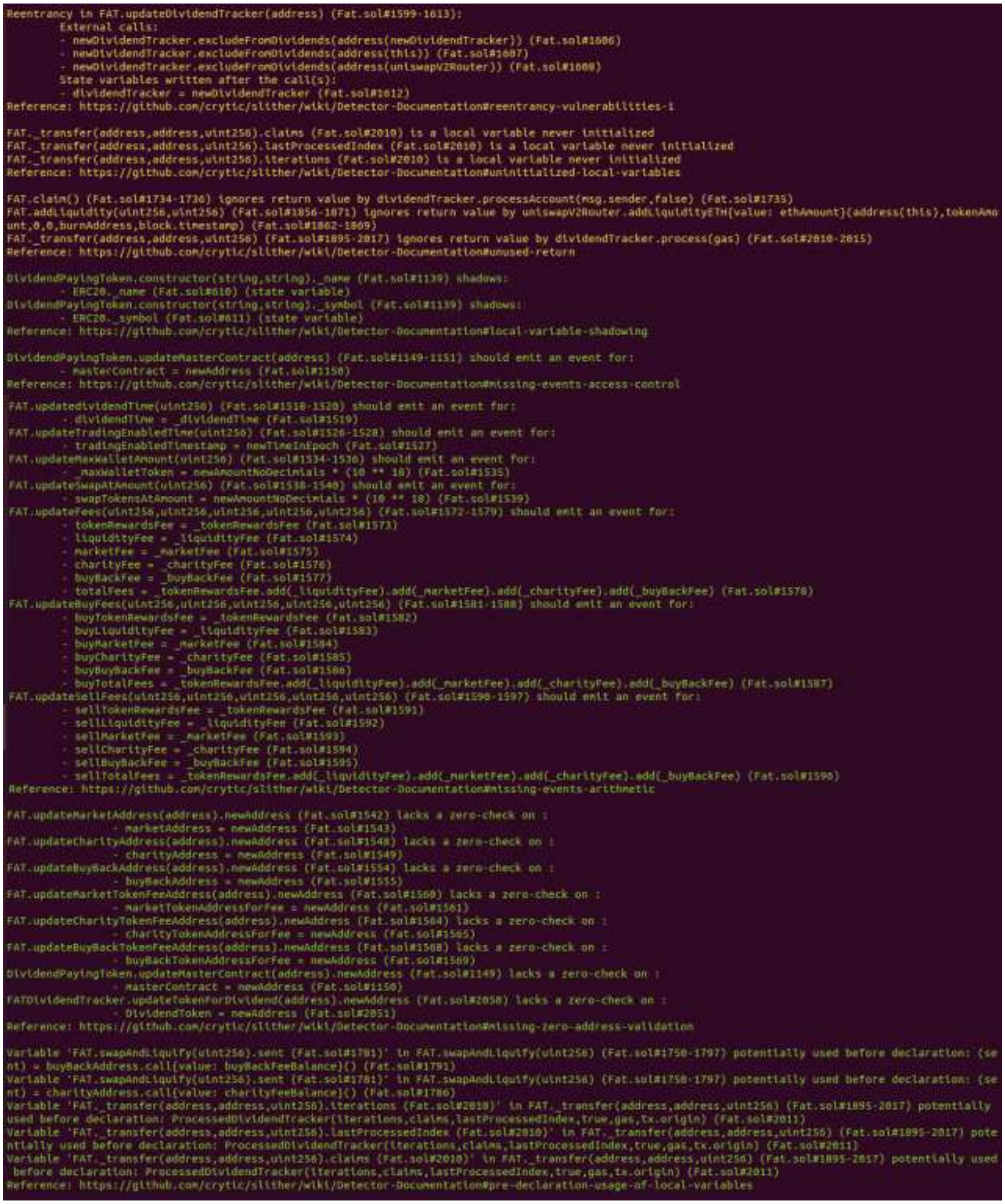
- Remediations: Developers should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, they may make use oracles.

Automated Tools Results

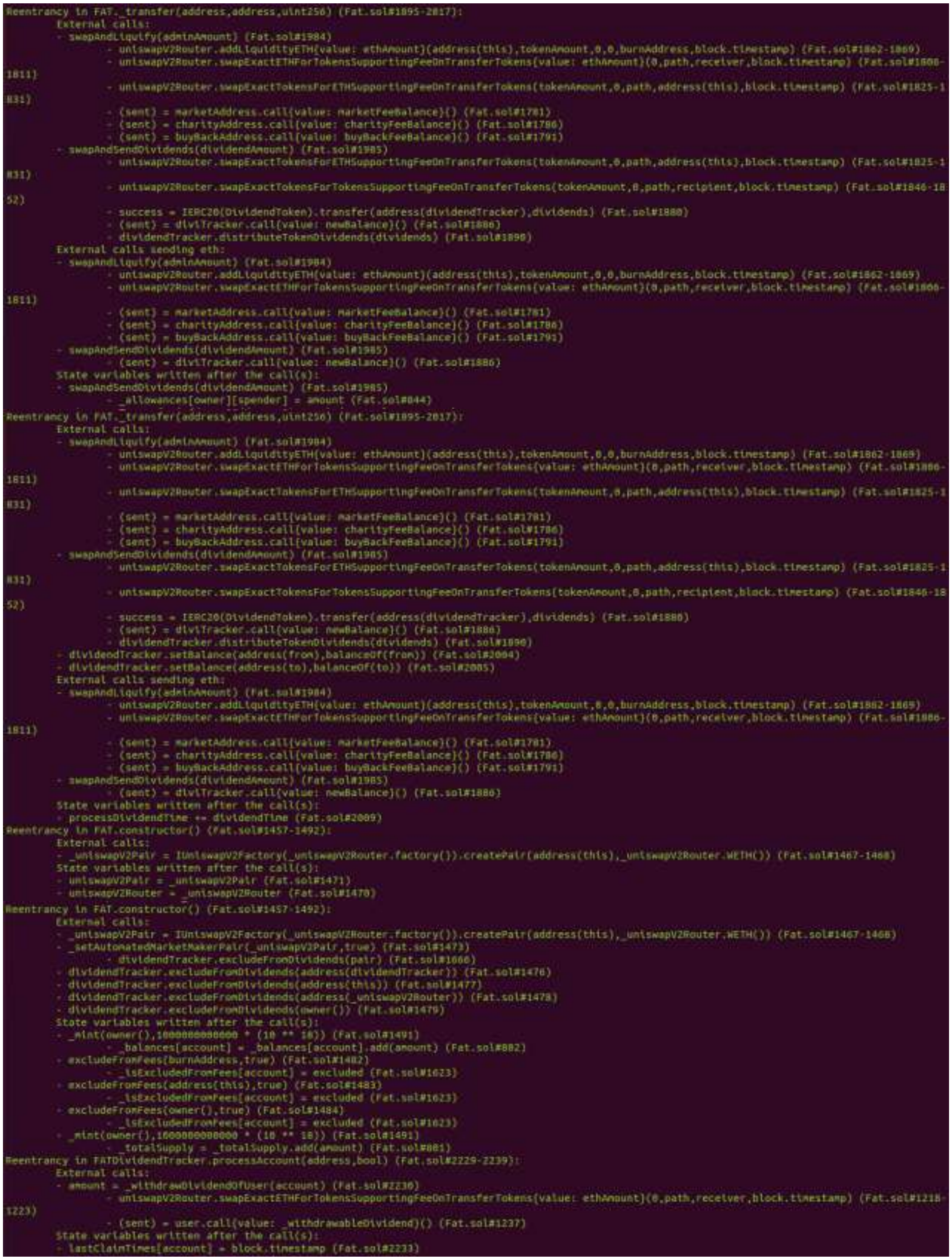
Slither: -



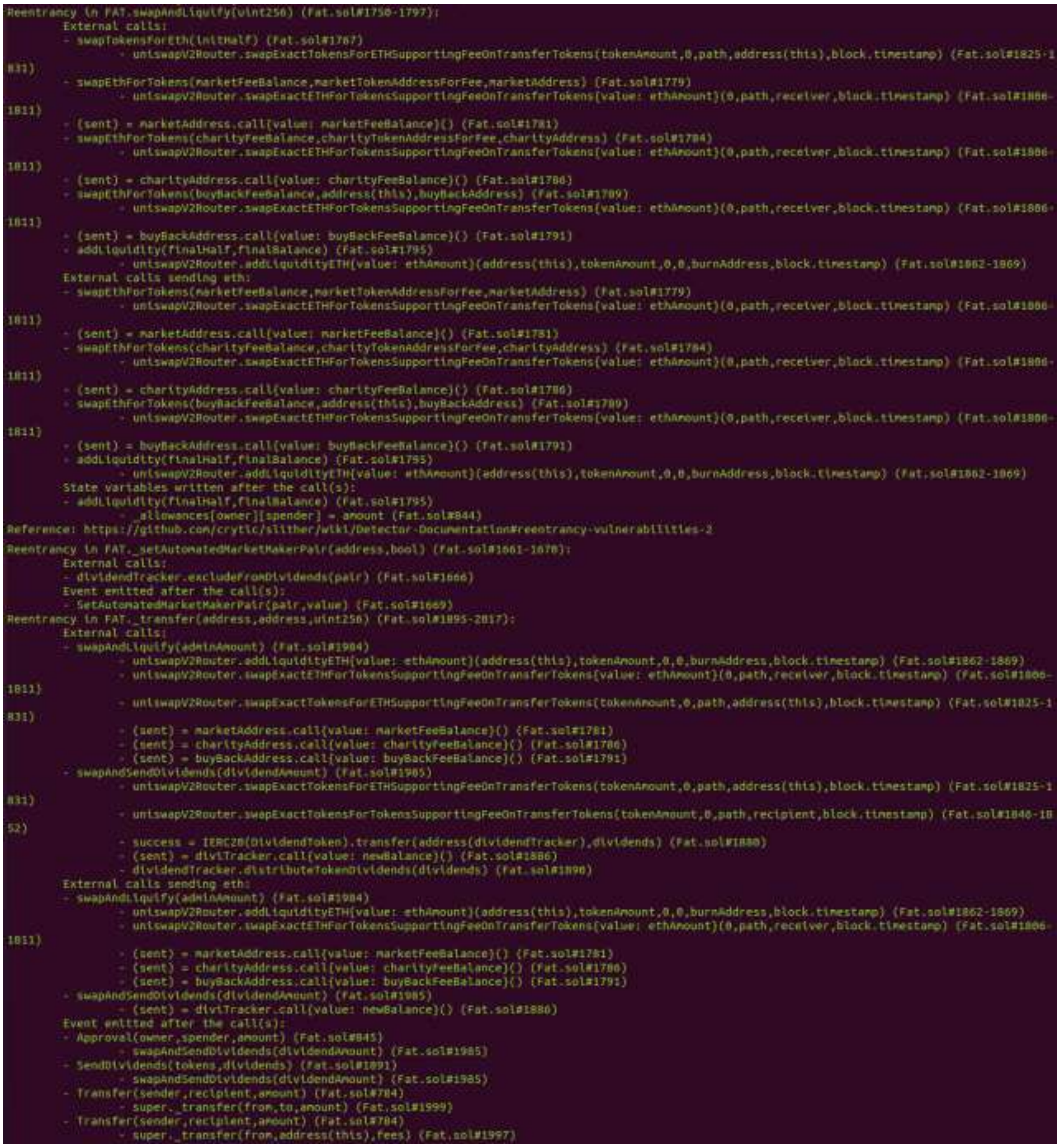
```
DividendPayingToken.swapEthForTokens(uint256,address,address) (Fat.sol#1211-1223) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1218-1223)
FAT.swapAndLiquify(uint256) (Fat.sol#1758-1797) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
FAT.swapEthForTokens(uint256,address,address) (Fat.sol#1799-1812) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-1811)
FAT.addLiquidity(uint256,uint256) (Fat.sol#1850-1871) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
FAT.swapAndSendDividends(uint256) (Fat.sol#1873-1893) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017):
        External calls:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
833)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
833)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1840-18
42)
                - success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
                - dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1889)
        External calls sending eth:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
        State variables written after the call(s):
        - super._transfer(from,address(this),fees) (Fat.sol#1997)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (Fat.sol#782)
                - _balances[recipient] = _balances[recipient].add(amount) (Fat.sol#783)
        - super._transfer(from,to,amount) (Fat.sol#1999)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (Fat.sol#782)
                - _balances[recipient] = _balances[recipient].add(amount) (Fat.sol#783)
        - restoreFees() (Fat.sol#2001)
                - buyBackFee = previousBuyBackFee (Fat.sol#1499)
        - restoreFees() (Fat.sol#2001)
                - charityFee = previousCharityFee (Fat.sol#1498)
        - restoreFees() (Fat.sol#2001)
                - liquidityFee = previousLiquidityFee (Fat.sol#1501)
        - restoreFees() (Fat.sol#2001)
                - marketFee = previousMarketFee (Fat.sol#1497)
        - swapping = false (Fat.sol#1986)
        - restoreFees() (Fat.sol#2001)
                - tokenRewardsFee = previousTokenRewardsFee (Fat.sol#1500)
        - restoreFees() (Fat.sol#2001)
                - totalFees = previousTotalFees (Fat.sol#1502)
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (Fat.sol#1230-1259):
        External calls:
        - (sent) = user.call{value: _withdrawableDividend}() (Fat.sol#1237)
        - success = swapEthForTokens(_withdrawableDividend,dividendTokenUser[user],user) (Fat.sol#1243)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1218-
1223)
        State variables written after the call(s):
        - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend) (Fat.sol#1251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```
]



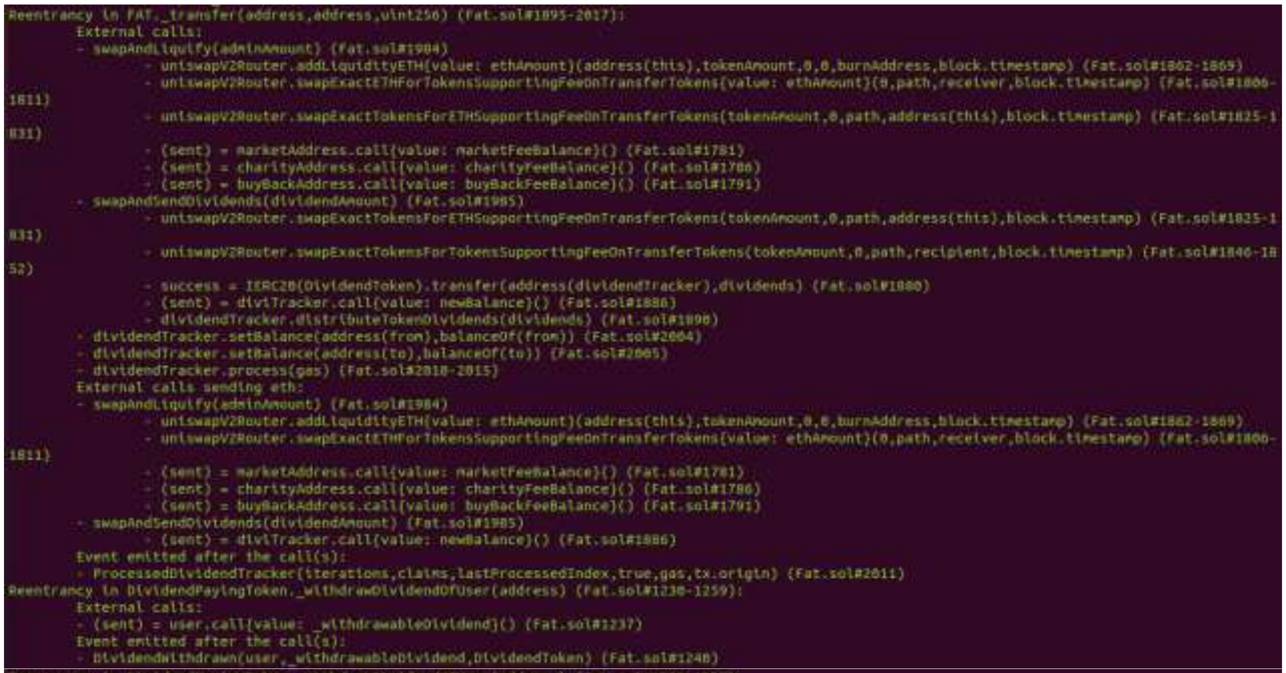
```
Reentrancy in FAT.updateDividendTracker(address) (Fat.sol#1599-1613):
        External calls:
        - newDividendTracker.excludeFromDividends(address(newDividendTracker)) (Fat.sol#1606)
        - newDividendTracker.excludeFromDividends(address(this)) (Fat.sol#1607)
        - newDividendTracker.excludeFromDividends(address(uniswapV2Router)) (Fat.sol#1608)
        State variables written after the call(s):
        - dividendTracker = newDividendTracker (Fat.sol#1612)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

FAT._transfer(address,address,uint256).claims (Fat.sol#2010) is a local variable never initialized
FAT._transfer(address,address,uint256).lastProcessedIndex (Fat.sol#2010) is a local variable never initialized
FAT._transfer(address,address,uint256).iterations (Fat.sol#2010) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

FAT.claim() (Fat.sol#1734-1736) ignores return value by dividendTracker.processAccount(msg.sender,false) (Fat.sol#1735)
FAT.addLiquidity(uint256,uint256) (Fat.sol#1856-1871) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmo
unt,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
FAT._transfer(address,address,uint256) (Fat.sol#1895-2017) ignores return value by dividendTracker.process(gas) (Fat.sol#2010-2015)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

DividendPayingToken.constructor(string,string)._name (Fat.sol#1139) shadows:
        - ERC20._name (Fat.sol#610) (state variable)
DividendPayingToken.constructor(string,string)._symbol (Fat.sol#1139) shadows:
        - ERC20._symbol (Fat.sol#611) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

DividendPayingToken.updateMasterContract(address) (Fat.sol#1149-1151) should emit an event for:
        - masterContract = newAddress (Fat.sol#1150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

FAT.updatedividendTime(uint256) (Fat.sol#1518-1520) should emit an event for:
        - dividendTime = _dividendTime (Fat.sol#1519)
FAT.updateTradingEnabledTime(uint256) (Fat.sol#1526-1528) should emit an event for:
        - tradingEnabledTimestamp = newTimeInEpoch (Fat.sol#1527)
FAT.updateMaxWalletAmount(uint256) (Fat.sol#1534-1536) should emit an event for:
        - _maxWalletToken = newAmountNoDecimials * (10 ** 18) (Fat.sol#1535)
FAT.updateSwapAtAmount(uint256) (Fat.sol#1538-1540) should emit an event for:
        - swapTokensAtAmount = newAmountNoDecimials * (10 ** 18) (Fat.sol#1539)
FAT.updateFees(uint256,uint256,uint256,uint256,uint256) (Fat.sol#1572-1579) should emit an event for:
        - tokenRewardsFee = _tokenRewardsFee (Fat.sol#1573)
        - liquidityFee = _liquidityFee (Fat.sol#1574)
        - marketFee = _marketFee (Fat.sol#1575)
        - charityFee = _charityFee (Fat.sol#1576)
        - buyBackFee = _buyBackFee (Fat.sol#1577)
        - totalFees = _tokenRewardsFee.add(_liquidityFee).add(_marketFee).add(_charityFee).add(_buyBackFee) (Fat.sol#1578)
FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256) (Fat.sol#1581-1588) should emit an event for:
        - buyTokenRewardsfee = _tokenRewardsfee (Fat.sol#1582)
        - buyLiquidityFee = _liquidityFee (Fat.sol#1583)
        - buyMarketFee = _marketFee (Fat.sol#1584)
        - buyCharityFee = _charityFee (Fat.sol#1585)
        - buyBuyBackFee = _buyBackFee (Fat.sol#1586)
        - buyTotalFees = _tokenRewardsFee.add(_liquidityFee).add(_marketFee).add(_charityFee).add(_buyBackFee) (Fat.sol#1587)
FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256) (Fat.sol#1590-1597) should emit an event for:
        - sellTokenRewardsFee = _tokenRewardsFee (Fat.sol#1591)
        - sellLiquidityFee = _liquidityFee (Fat.sol#1592)
        - sellMarketFee = _marketFee (Fat.sol#1593)
        - sellCharityFee = _charityFee (Fat.sol#1594)
        - sellBuyBackFee = _buyBackFee (Fat.sol#1595)
        - sellTotalFees = _tokenRewardsFee.add(_liquidityFee).add(_marketFee).add(_charityFee).add(_buyBackFee) (Fat.sol#1596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

FAT.updateMarketAddress(address).newAddress (Fat.sol#1542) lacks a zero-check on :
                - marketAddress = newAddress (Fat.sol#1543)
FAT.updateCharityAddress(address).newAddress (Fat.sol#1548) lacks a zero-check on :
                - charityAddress = newAddress (Fat.sol#1549)
FAT.updateBuyBackAddress(address).newAddress (Fat.sol#1554) lacks a zero-check on :
                - buyBackAddress = newAddress (Fat.sol#1555)
FAT.updateMarketTokenFeeAddress(address).newAddress (Fat.sol#1560) lacks a zero-check on :
                - marketTokenAddressForFee = newAddress (Fat.sol#1561)
FAT.updateCharityTokenFeeAddress(address).newAddress (Fat.sol#1564) lacks a zero-check on :
                - charityTokenAddressForFee = newAddress (Fat.sol#1565)
FAT.updateBuyBackTokenFeeAddress(address).newAddress (Fat.sol#1568) lacks a zero-check on :
                - buyBackTokenAddressForFee = newAddress (Fat.sol#1569)
DividendPayingToken.updateMasterContract(address).newAddress (Fat.sol#1149) lacks a zero-check on :
                - masterContract = newAddress (Fat.sol#1150)
FATDividendTracker.updateTokenForDividend(address).newAddress (Fat.sol#2050) lacks a zero-check on :
                - DividendToken = newAddress (Fat.sol#2051)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'FAT.swapAndLiquify(uint256).sent (Fat.sol#1781)' in FAT.swapAndLiquify(uint256) (Fat.sol#1750-1797) potentially used before declaration: (se
nt) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
Variable 'FAT.swapAndLiquify(uint256).sent (Fat.sol#1781)' in FAT.swapAndLiquify(uint256) (Fat.sol#1750-1797) potentially used before declaration: (se
nt) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
Variable 'FAT._transfer(address,address,uint256).iterations (Fat.sol#2010)' in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017) potentially
used before declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (Fat.sol#2011)
Variable 'FAT._transfer(address,address,uint256).lastProcessedIndex (Fat.sol#2010)' in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017) pote
ntially used before declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (Fat.sol#2011)
Variable 'FAT._transfer(address,address,uint256).claims (Fat.sol#2010)' in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017) potentially used
 before declaration: ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (Fat.sol#2011)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



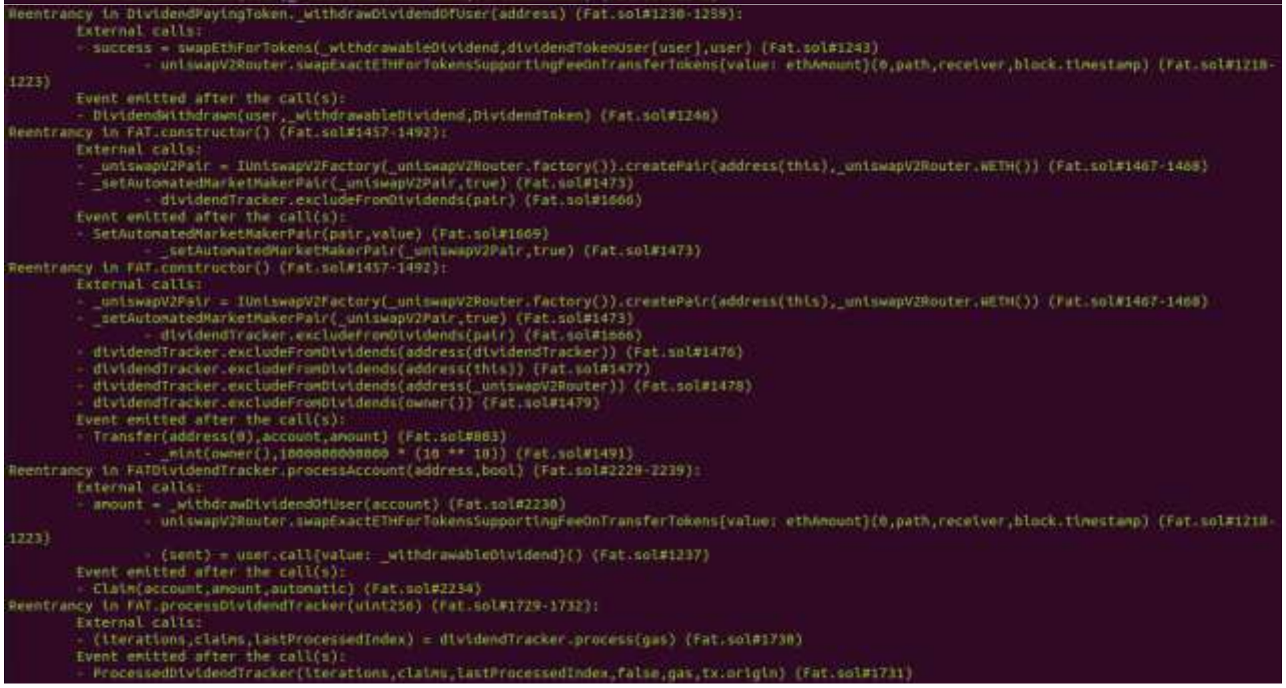
```
Reentrancy in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017):
	External calls:
	- swapAndLiquify(adminAmount) (Fat.sol#1984)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
		- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
		- (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
		- (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
		- (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
	- swapAndSendDividends(dividendAmount) (Fat.sol#1985)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
		- uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1846-18
52)
		- success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
		- (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
		- dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1890)
	External calls sending eth:
	- swapAndLiquify(adminAmount) (Fat.sol#1984)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
		- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
		- (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
		- (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
		- (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
	- swapAndSendDividends(dividendAmount) (Fat.sol#1985)
		- (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
	State variables written after the call(s):
	- swapAndSendDividends(dividendAmount) (Fat.sol#1985)
		- _allowances[owner][spender] = amount (Fat.sol#844)
Reentrancy in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017):
	External calls:
	- swapAndLiquify(adminAmount) (Fat.sol#1984)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
		- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
		- (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
		- (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
		- (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
	- swapAndSendDividends(dividendAmount) (Fat.sol#1985)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
		- uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1846-18
52)
		- success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
		- (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
		- dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1890)
	- dividendTracker.setBalance(address(from),balanceOf(from)) (Fat.sol#2004)
	- dividendTracker.setBalance(address(to),balanceOf(to)) (Fat.sol#2005)
	External calls sending eth:
	- swapAndLiquify(adminAmount) (Fat.sol#1984)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
		- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
		- (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
		- (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
		- (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
	- swapAndSendDividends(dividendAmount) (Fat.sol#1985)
		- (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
	State variables written after the call(s):
	- processDividendTime += dividendTime (Fat.sol#2009)
Reentrancy in FAT.constructor() (Fat.sol#1457-1492):
	External calls:
	- _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Fat.sol#1467-1468)
	State variables written after the call(s):
	- uniswapV2Pair = _uniswapV2Pair (Fat.sol#1471)
	- uniswapV2Router = _uniswapV2Router (Fat.sol#1470)
Reentrancy in FAT.constructor() (Fat.sol#1457-1492):
	External calls:
	- _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Fat.sol#1467-1468)
	- _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Fat.sol#1473)
		- dividendTracker.excludeFromDividends(pair) (Fat.sol#1666)
	- dividendTracker.excludeFromDividends(address(dividendTracker)) (Fat.sol#1476)
	- dividendTracker.excludeFromDividends(address(this)) (Fat.sol#1477)
	- dividendTracker.excludeFromDividends(address(_uniswapV2Router)) (Fat.sol#1478)
	- dividendTracker.excludeFromDividends(owner()) (Fat.sol#1479)
	State variables written after the call(s):
	- _mint(owner(),1000000000000 * (10 ** 18)) (Fat.sol#1491)
		- _balances[account] = _balances[account].add(amount) (Fat.sol#882)
	- excludeFromFees(burnAddress,true) (Fat.sol#1482)
		- _isExcludedFromFees[account] = excluded (Fat.sol#1623)
	- excludeFromFees(address(this),true) (Fat.sol#1483)
		- _isExcludedFromFees[account] = excluded (Fat.sol#1623)
	- excludeFromFees(owner(),true) (Fat.sol#1484)
		- _isExcludedFromFees[account] = excluded (Fat.sol#1623)
	- _mint(owner(),1000000000000 * (10 ** 18)) (Fat.sol#1491)
		- _totalSupply = _totalSupply.add(amount) (Fat.sol#881)
Reentrancy in FATDividendTracker.processAccount(address,bool) (Fat.sol#2229-2239):
	External calls:
	- amount = _withdrawDividendOfUser(account) (Fat.sol#2230)
		- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1218-
1223)
		- (sent) = user.call{value: _withdrawableDividend}() (Fat.sol#1237)
	State variables written after the call(s):
	- lastClaimTimes[account] = block.timestamp (Fat.sol#2233)
```



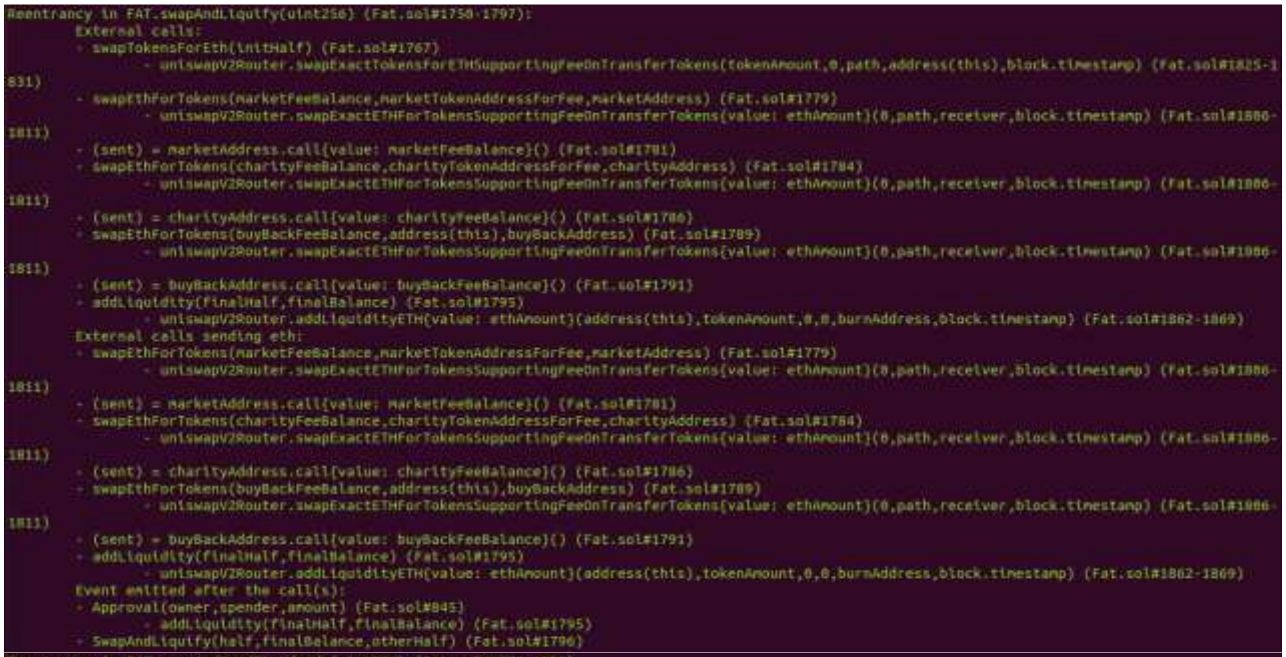
```
Reentrancy in FAT.swapAndLiquify(uint256) (Fat.sol#1750-1797):
        External calls:
        - swapTokensForEth(initHalf) (Fat.sol#1767)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
        - swapEthForTokens(marketFeeBalance,marketTokenAddressForFee,marketAddress) (Fat.sol#1779)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - swapEthForTokens(charityFeeBalance,charityTokenAddressForFee,charityAddress) (Fat.sol#1784)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - swapEthForTokens(buyBackFeeBalance,address(this),buyBackAddress) (Fat.sol#1789)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
        External calls sending eth:
        - swapEthForTokens(marketFeeBalance,marketTokenAddressForFee,marketAddress) (Fat.sol#1779)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - swapEthForTokens(charityFeeBalance,charityTokenAddressForFee,charityAddress) (Fat.sol#1784)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - swapEthForTokens(buyBackFeeBalance,address(this),buyBackAddress) (Fat.sol#1789)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
        State variables written after the call(s):
        - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
                - _allowances[owner][spender] = amount (Fat.sol#844)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
Reentrancy in FAT._setAutomatedMarketMakerPair(address,bool) (Fat.sol#1661-1670):
        External calls:
        - dividendTracker.excludeFromDividends(pair) (Fat.sol#1666)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (Fat.sol#1669)
Reentrancy in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017):
        External calls:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1846-18
52)
                - success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
                - dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1890)
        External calls sending eth:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Fat.sol#845)
                - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
        - SendDividends(tokens,dividends) (Fat.sol#1891)
                - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
        - Transfer(sender,recipient,amount) (Fat.sol#784)
                - super._transfer(from,to,amount) (Fat.sol#1999)
        - Transfer(sender,recipient,amount) (Fat.sol#784)
                - super._transfer(from,address(this),fees) (Fat.sol#1997)
```



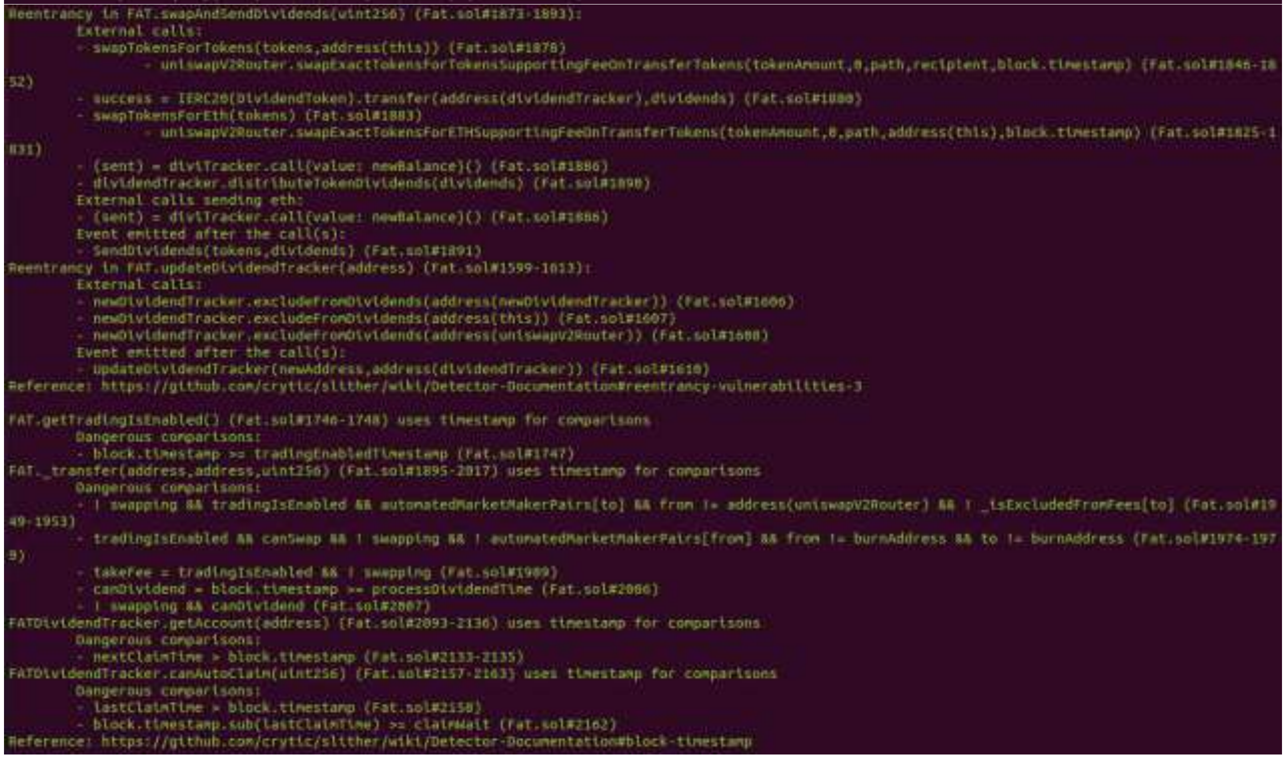
```
Reentrancy in FAT._transfer(address,address,uint256) (Fat.sol#1895-2017):
        External calls:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1846-18
52)
                - success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
                - dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1890)
        - dividendTracker.setBalance(address(from),balanceOf(from)) (Fat.sol#2004)
        - dividendTracker.setBalance(address(to),balanceOf(to)) (Fat.sol#2005)
        - dividendTracker.process(gas) (Fat.sol#2010-2015)
        External calls sending eth:
        - swapAndLiquify(adminAmount) (Fat.sol#1984)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
                - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
                - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
                - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - swapAndSendDividends(dividendAmount) (Fat.sol#1985)
                - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (Fat.sol#2011)
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (Fat.sol#1230-1259):
        External calls:
        - (sent) = user.call{value: _withdrawableDividend}() (Fat.sol#1237)
        Event emitted after the call(s):
        - DividendWithdrawn(user,_withdrawableDividend,DividendToken) (Fat.sol#1248)
```



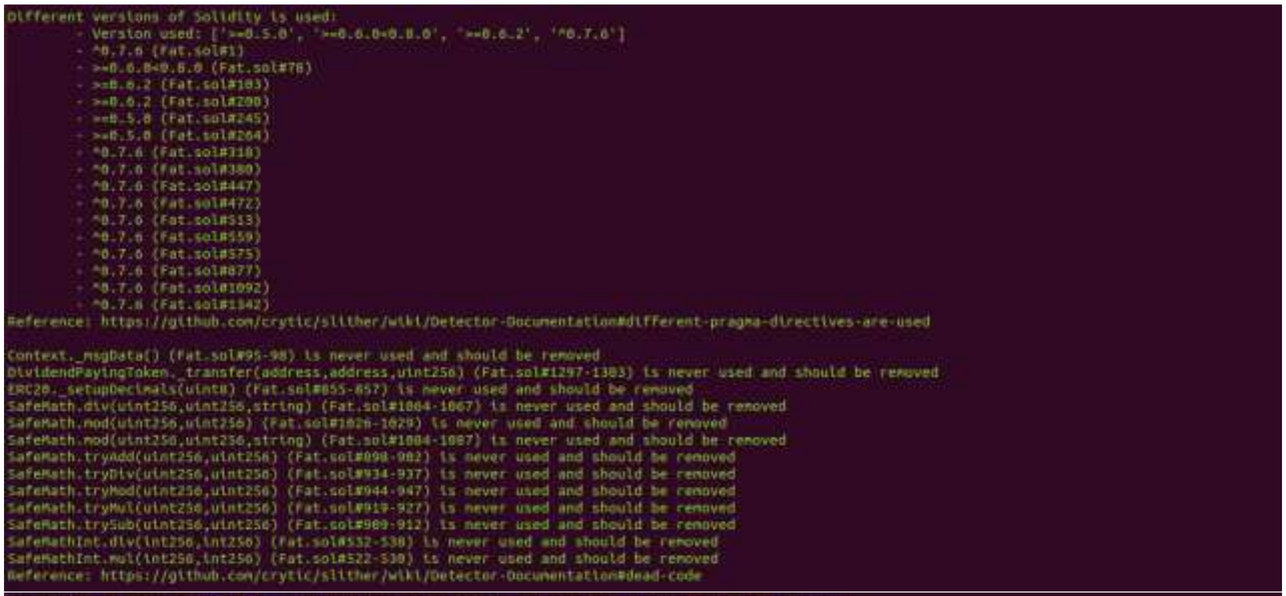
```
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (Fat.sol#1230-1259):
        External calls:
        - success = swapEthForTokens(_withdrawableDividend,dividendTokenUser[user],user) (Fat.sol#1243)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1218-
1223)
        Event emitted after the call(s):
        - DividendWithdrawn(user,_withdrawableDividend,DividendToken) (Fat.sol#1248)
Reentrancy in FAT.constructor() (Fat.sol#1457-1492):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Fat.sol#1467-1468)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Fat.sol#1473)
                - dividendTracker.excludeFromDividends(pair) (Fat.sol#1666)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (Fat.sol#1669)
                - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Fat.sol#1473)
Reentrancy in FAT.constructor() (Fat.sol#1457-1492):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Fat.sol#1467-1468)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (Fat.sol#1473)
                - dividendTracker.excludeFromDividends(pair) (Fat.sol#1666)
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (Fat.sol#1476)
        - dividendTracker.excludeFromDividends(address(this)) (Fat.sol#1477)
        - dividendTracker.excludeFromDividends(address(_uniswapV2Router)) (Fat.sol#1478)
        - dividendTracker.excludeFromDividends(owner()) (Fat.sol#1479)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Fat.sol#863)
                - _mint(owner(),1000000000000 * (10 ** 18)) (Fat.sol#1491)
Reentrancy in FATDividendTracker.processAccount(address,bool) (Fat.sol#2229-2239):
        External calls:
        - amount = _withdrawDividendOfUser(account) (Fat.sol#2230)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1218-
1223)
                - (sent) = user.call{value: _withdrawableDividend}() (Fat.sol#1237)
        Event emitted after the call(s):
        - Claim(account,amount,automatic) (Fat.sol#2234)
Reentrancy in FAT.processDividendTracker(uint256) (Fat.sol#1729-1732):
        External calls:
        - (iterations,claims,lastProcessedIndex) = dividendTracker.process(gas) (Fat.sol#1730)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,false,gas,tx.origin) (Fat.sol#1731)
```



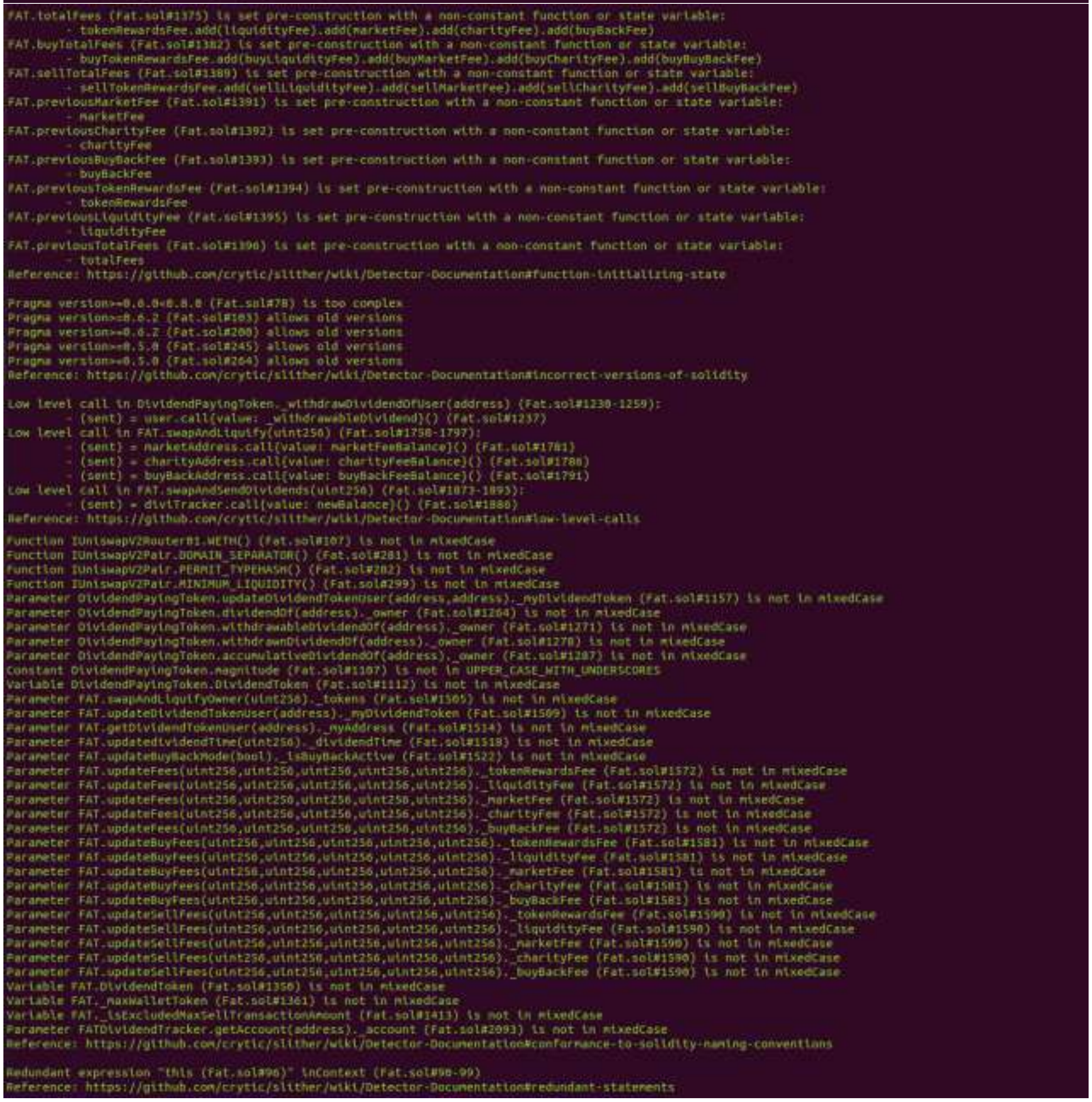
```
Reentrancy in FAT.swapAndLiquify(uint256) (Fat.sol#1750-1797):
        External calls:
        - swapTokensForEth(initHalf) (Fat.sol#1767)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
        - swapEthForTokens(marketFeeBalance,marketTokenAddressForFee,marketAddress) (Fat.sol#1779)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - swapEthForTokens(charityFeeBalance,charityTokenAddressForFee,charityAddress) (Fat.sol#1784)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - swapEthForTokens(buyBackFeeBalance,address(this),buyBackAddress) (Fat.sol#1789)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
        External calls sending eth:
        - swapEthForTokens(marketFeeBalance,marketTokenAddressForFee,marketAddress) (Fat.sol#1779)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - swapEthForTokens(charityFeeBalance,charityTokenAddressForFee,charityAddress) (Fat.sol#1784)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - swapEthForTokens(buyBackFeeBalance,address(this),buyBackAddress) (Fat.sol#1789)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,receiver,block.timestamp) (Fat.sol#1806-
1811)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
        - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,burnAddress,block.timestamp) (Fat.sol#1862-1869)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Fat.sol#845)
                - addLiquidity(finalHalf,finalBalance) (Fat.sol#1795)
        - SwapAndLiquify(half,finalBalance,otherHalf) (Fat.sol#1796)
```



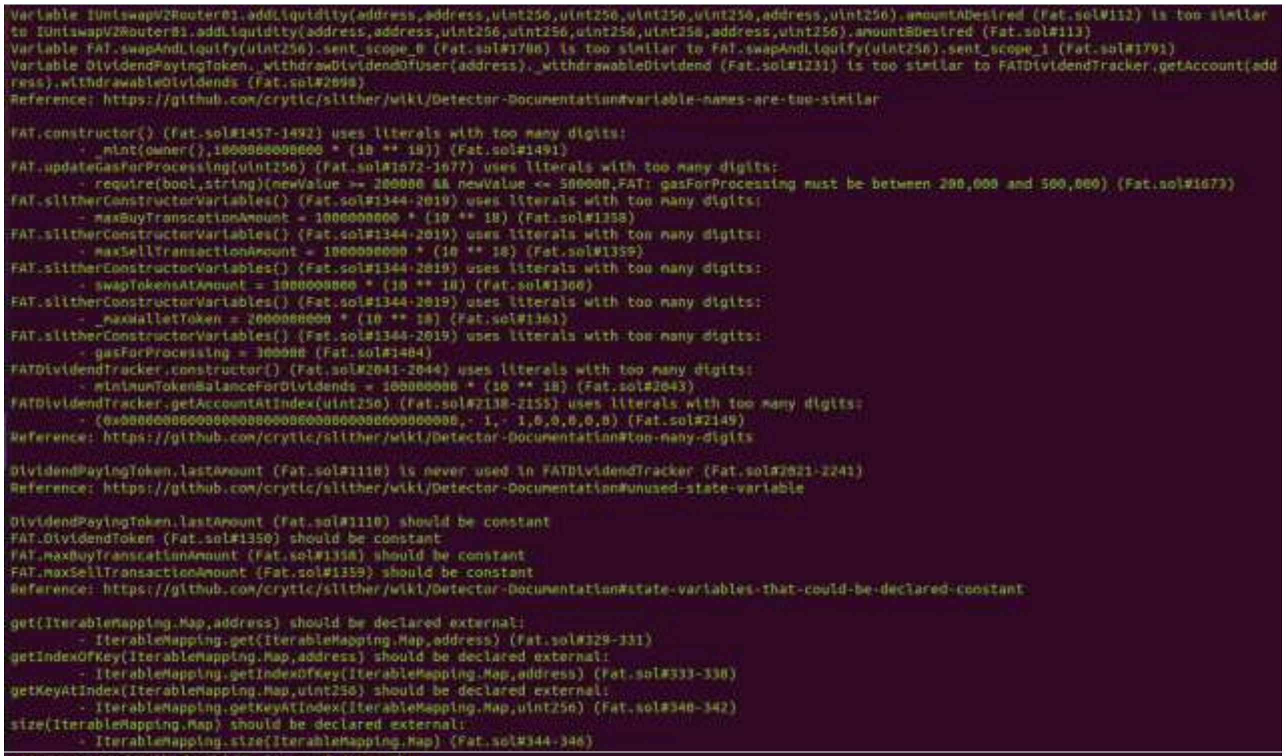
```
Reentrancy in FAT.swapAndSendDividends(uint256) (Fat.sol#1873-1893):
        External calls:
        - swapTokensForTokens(tokens,address(this)) (Fat.sol#1878)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,recipient,block.timestamp) (Fat.sol#1846-18
52)
        - success = IERC20(DividendToken).transfer(address(dividendTracker),dividends) (Fat.sol#1880)
        - swapTokensForEth(tokens) (Fat.sol#1883)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fat.sol#1825-1
831)
        - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
        - dividendTracker.distributeTokenDividends(dividends) (Fat.sol#1890)
        External calls sending eth:
        - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
        Event emitted after the call(s):
        - SendDividends(tokens,dividends) (Fat.sol#1891)
Reentrancy in FAT.updateDividendTracker(address) (Fat.sol#1599-1613):
        External calls:
        - newDividendTracker.excludeFromDividends(address(newDividendTracker)) (Fat.sol#1606)
        - newDividendTracker.excludeFromDividends(address(this)) (Fat.sol#1607)
        - newDividendTracker.excludeFromDividends(address(uniswapV2Router)) (Fat.sol#1608)
        Event emitted after the call(s):
        - UpdateDividendTracker(newAddress,address(dividendTracker)) (Fat.sol#1610)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

FAT.getTradingIsEnabled() (Fat.sol#1746-1748) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= tradingEnabledTimestamp (Fat.sol#1747)
FAT._transfer(address,address,uint256) (Fat.sol#1895-2017) uses timestamp for comparisons
        Dangerous comparisons:
        - ! swapping && tradingIsEnabled && automatedMarketMakerPairs[to] && from != address(uniswapV2Router) && ! _isExcludedFromFees[to] (Fat.sol#19
49-1953)
        - tradingIsEnabled && canSwap && ! swapping && ! automatedMarketMakerPairs[from] && from != burnAddress && to != burnAddress (Fat.sol#1974-197
9)
        - takeFee = tradingIsEnabled && ! swapping (Fat.sol#1989)
        - canDividend = block.timestamp >= processDividendTime (Fat.sol#2006)
        - ! swapping && canDividend (Fat.sol#2007)
FATDividendTracker.getAccount(address) (Fat.sol#2093-2136) uses timestamp for comparisons
        Dangerous comparisons:
        - nextClaimTime > block.timestamp (Fat.sol#2133-2135)
FATDividendTracker.canAutoClaim(uint256) (Fat.sol#2157-2163) uses timestamp for comparisons
        Dangerous comparisons:
        - lastClaimTime > block.timestamp (Fat.sol#2158)
        - block.timestamp.sub(lastClaimTime) >= claimWait (Fat.sol#2162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```



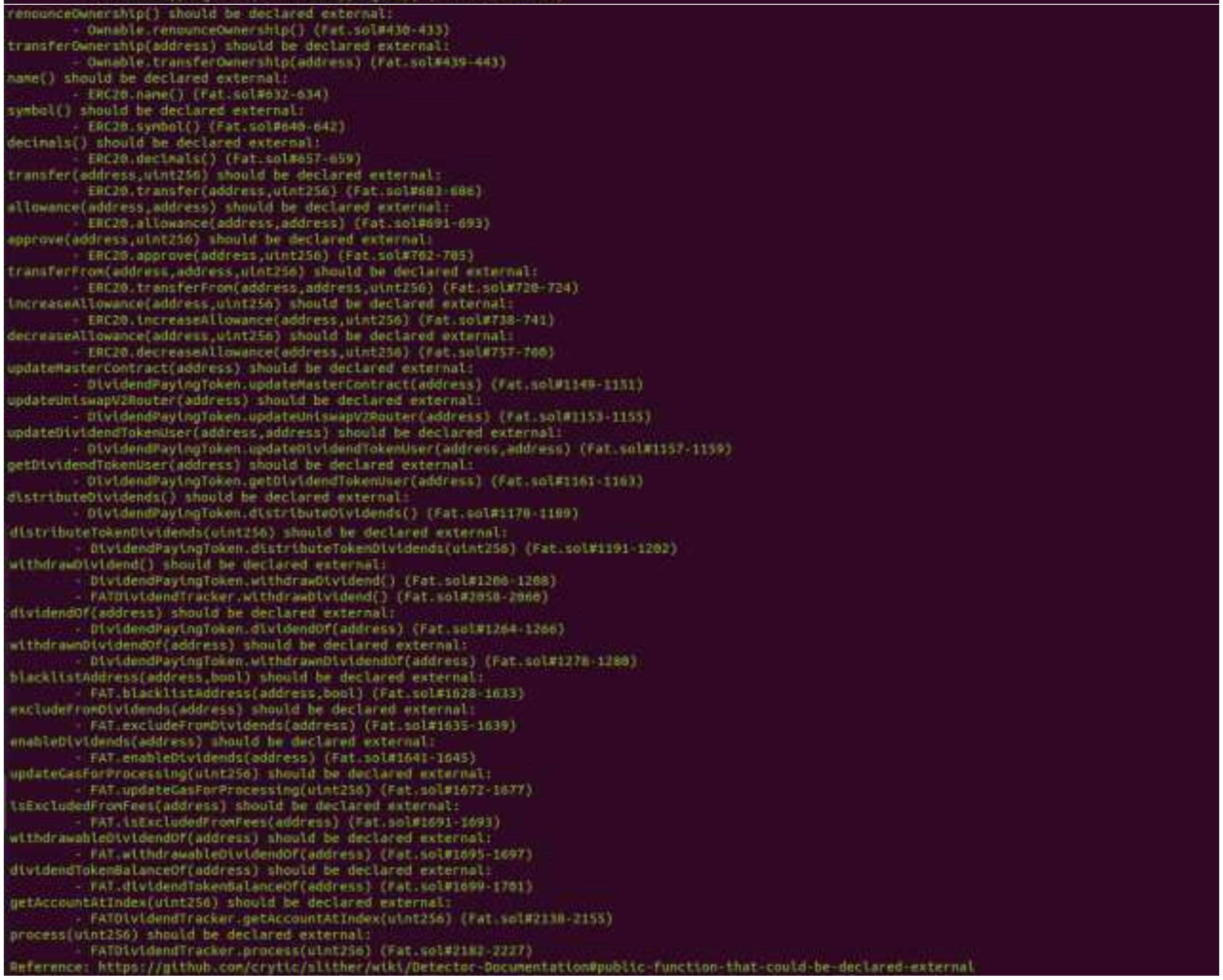
```
Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.0<0.8.0', '>=0.6.2', '^0.7.6']
        - ^0.7.6 (Fat.sol#1)
        - >=0.6.0<0.8.0 (Fat.sol#78)
        - >=0.6.2 (Fat.sol#103)
        - >=0.6.2 (Fat.sol#200)
        - >=0.5.0 (Fat.sol#245)
        - >=0.5.0 (Fat.sol#264)
        - ^0.7.6 (Fat.sol#318)
        - ^0.7.6 (Fat.sol#380)
        - ^0.7.6 (Fat.sol#447)
        - ^0.7.6 (Fat.sol#472)
        - ^0.7.6 (Fat.sol#513)
        - ^0.7.6 (Fat.sol#559)
        - ^0.7.6 (Fat.sol#575)
        - ^0.7.6 (Fat.sol#877)
        - ^0.7.6 (Fat.sol#1092)
        - ^0.7.6 (Fat.sol#1342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (Fat.sol#95-98) is never used and should be removed
DividendPayingToken._transfer(address,address,uint256) (Fat.sol#1297-1303) is never used and should be removed
ERC20._setupDecimals(uint8) (Fat.sol#855-857) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Fat.sol#1064-1067) is never used and should be removed
SafeMath.mod(uint256,uint256) (Fat.sol#1026-1029) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Fat.sol#1084-1087) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (Fat.sol#898-902) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (Fat.sol#934-937) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (Fat.sol#944-947) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (Fat.sol#919-927) is never used and should be removed
SafeMath.trySub(uint256,uint256) (Fat.sol#909-912) is never used and should be removed
SafeMathInt.div(int256,int256) (Fat.sol#532-538) is never used and should be removed
SafeMathInt.mul(int256,int256) (Fat.sol#522-530) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



```
FAT.totalFees (Fat.sol#1375) is set pre-construction with a non-constant function or state variable:
        - tokenRewardsFee.add(liquidityFee).add(marketFee).add(charityFee).add(buyBackFee)
FAT.buyTotalFees (Fat.sol#1382) is set pre-construction with a non-constant function or state variable:
        - buyTokenRewardsFee.add(buyLiquidityFee).add(buyMarketFee).add(buyCharityFee).add(buyBuyBackFee)
FAT.sellTotalFees (Fat.sol#1389) is set pre-construction with a non-constant function or state variable:
        - sellTokenRewardsFee.add(sellLiquidityFee).add(sellMarketFee).add(sellCharityFee).add(sellBuyBackFee)
FAT.previousMarketFee (Fat.sol#1391) is set pre-construction with a non-constant function or state variable:
        - marketFee
FAT.previousCharityFee (Fat.sol#1392) is set pre-construction with a non-constant function or state variable:
        - charityFee
FAT.previousBuyBackFee (Fat.sol#1393) is set pre-construction with a non-constant function or state variable:
        - buyBackFee
FAT.previousTokenRewardsFee (Fat.sol#1394) is set pre-construction with a non-constant function or state variable:
        - tokenRewardsFee
FAT.previousLiquidityFee (Fat.sol#1395) is set pre-construction with a non-constant function or state variable:
        - liquidityFee
FAT.previousTotalFees (Fat.sol#1396) is set pre-construction with a non-constant function or state variable:
        - totalFees
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version>=0.6.0<0.8.0 (Fat.sol#78) is too complex
Pragma version>=0.6.2 (Fat.sol#103) allows old versions
Pragma version>=0.6.2 (Fat.sol#200) allows old versions
Pragma version>=0.5.0 (Fat.sol#245) allows old versions
Pragma version>=0.5.0 (Fat.sol#264) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in DividendPayingToken._withdrawDividendOfUser(address) (Fat.sol#1230-1259):
        - (sent) = user.call{value: _withdrawableDividend}() (Fat.sol#1237)
Low level call in FAT.swapAndLiquify(uint256) (Fat.sol#1758-1797):
        - (sent) = marketAddress.call{value: marketFeeBalance}() (Fat.sol#1781)
        - (sent) = charityAddress.call{value: charityFeeBalance}() (Fat.sol#1786)
        - (sent) = buyBackAddress.call{value: buyBackFeeBalance}() (Fat.sol#1791)
Low level call in FAT.swapAndSendDividends(uint256) (Fat.sol#1873-1893):
        - (sent) = diviTracker.call{value: newBalance}() (Fat.sol#1886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Router01.WETH() (Fat.sol#107) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Fat.sol#281) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Fat.sol#282) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Fat.sol#299) is not in mixedCase
Parameter DividendPayingToken.updateDividendTokenUser(address,address)._myDividendToken (Fat.sol#1157) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (Fat.sol#1264) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (Fat.sol#1271) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (Fat.sol#1278) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (Fat.sol#1287) is not in mixedCase
Constant DividendPayingToken.magnitude (Fat.sol#1107) is not in UPPER_CASE_WITH_UNDERSCORES
Variable DividendPayingToken.DividendToken (Fat.sol#1112) is not in mixedCase
Parameter FAT.swapAndLiquifyOwner(uint256)._tokens (Fat.sol#1505) is not in mixedCase
Parameter FAT.updateDividendTokenUser(address)._myDividendToken (Fat.sol#1509) is not in mixedCase
Parameter FAT.getDividendTokenUser(address)._myAddress (Fat.sol#1514) is not in mixedCase
Parameter FAT.updatedividendTime(uint256)._dividendTime (Fat.sol#1518) is not in mixedCase
Parameter FAT.updateBuyBackMode(bool)._isBuyBackActive (Fat.sol#1522) is not in mixedCase
Parameter FAT.updateFees(uint256,uint256,uint256,uint256,uint256)._tokenRewardsFee (Fat.sol#1572) is not in mixedCase
Parameter FAT.updateFees(uint256,uint256,uint256,uint256,uint256)._liquidityFee (Fat.sol#1572) is not in mixedCase
Parameter FAT.updateFees(uint256,uint256,uint256,uint256,uint256)._marketFee (Fat.sol#1572) is not in mixedCase
Parameter FAT.updateFees(uint256,uint256,uint256,uint256,uint256)._charityFee (Fat.sol#1572) is not in mixedCase
Parameter FAT.updateFees(uint256,uint256,uint256,uint256,uint256)._buyBackFee (Fat.sol#1572) is not in mixedCase
Parameter FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._tokenRewardsFee (Fat.sol#1581) is not in mixedCase
Parameter FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._liquidityFee (Fat.sol#1581) is not in mixedCase
Parameter FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._marketFee (Fat.sol#1581) is not in mixedCase
Parameter FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._charityFee (Fat.sol#1581) is not in mixedCase
Parameter FAT.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._buyBackFee (Fat.sol#1581) is not in mixedCase
Parameter FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256)._tokenRewardsFee (Fat.sol#1590) is not in mixedCase
Parameter FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256)._liquidityFee (Fat.sol#1590) is not in mixedCase
Parameter FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256)._marketFee (Fat.sol#1590) is not in mixedCase
Parameter FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256)._charityFee (Fat.sol#1590) is not in mixedCase
Parameter FAT.updateSellFees(uint256,uint256,uint256,uint256,uint256)._buyBackFee (Fat.sol#1590) is not in mixedCase
Variable FAT.DividendToken (Fat.sol#1350) is not in mixedCase
Variable FAT._maxWalletToken (Fat.sol#1361) is not in mixedCase
Variable FAT._isExcludedMaxSellTransactionAmount (Fat.sol#1413) is not in mixedCase
Parameter FATDividendTracker.getAccount(address)._account (Fat.sol#2093) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Fat.sol#96)" inContext (Fat.sol#90-99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

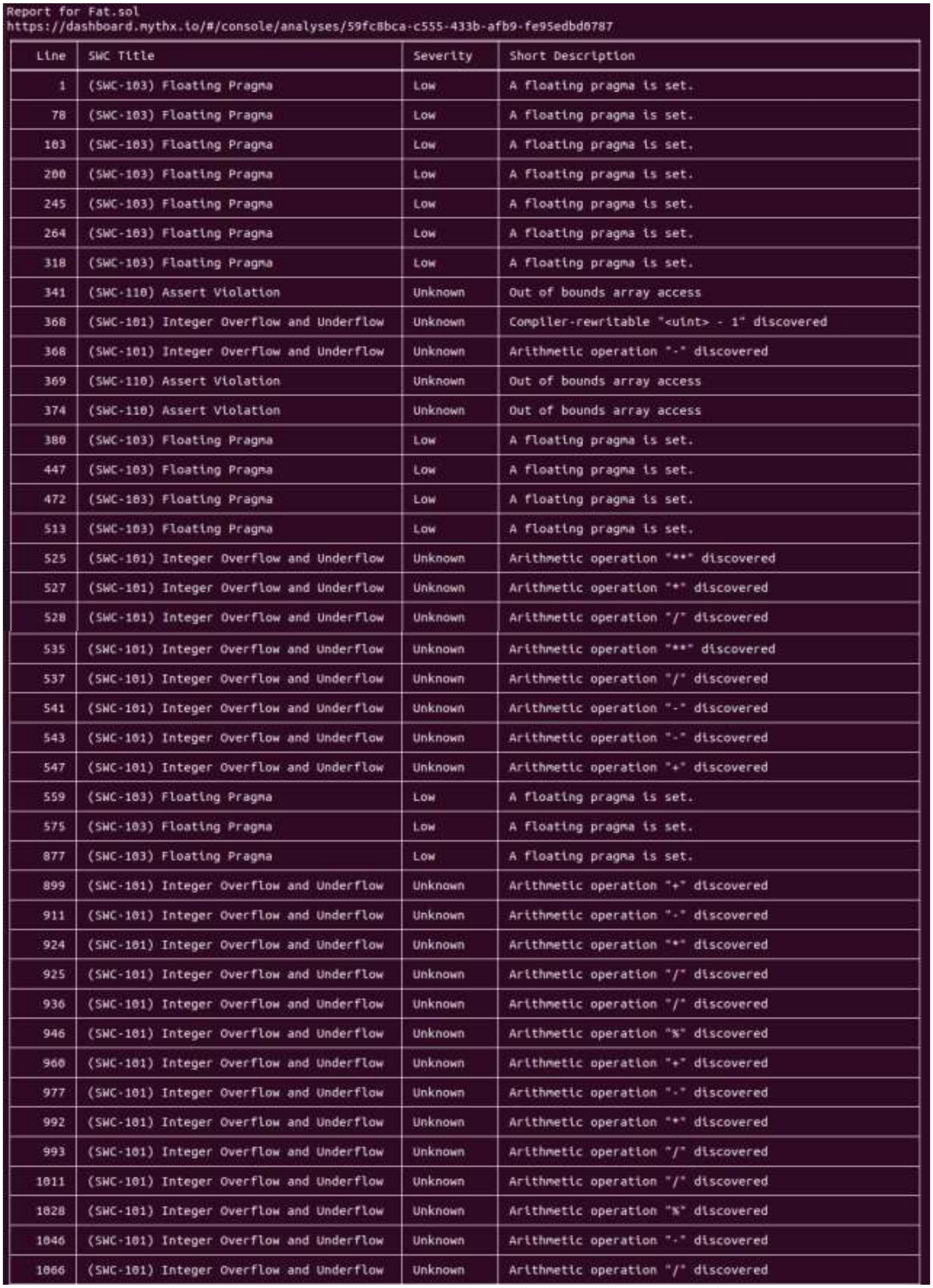


```
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Fat.sol#112) is too similar
to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Fat.sol#113)
Variable FAT.swapAndLiquify(uint256).sent_scope_0 (Fat.sol#1786) is too similar to FAT.swapAndLiquify(uint256).sent_scope_1 (Fat.sol#1791)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (Fat.sol#1231) is too similar to FATDividendTracker.getAccount(add
ress).withdrawableDividends (Fat.sol#2098)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

FAT.constructor() (Fat.sol#1457-1492) uses literals with too many digits:
        - _mint(owner(),1000000000000 * (10 ** 18)) (Fat.sol#1491)
FAT.updateGasForProcessing(uint256) (Fat.sol#1672-1677) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,FAT: gasForProcessing must be between 200,000 and 500,000) (Fat.sol#1673)
FAT.slitherConstructorVariables() (Fat.sol#1344-2019) uses literals with too many digits:
        - maxBuyTranscationAmount = 1000000000 * (10 ** 18) (Fat.sol#1358)
FAT.slitherConstructorVariables() (Fat.sol#1344-2019) uses literals with too many digits:
        - maxSellTransactionAmount = 1000000000 * (10 ** 18) (Fat.sol#1359)
FAT.slitherConstructorVariables() (Fat.sol#1344-2019) uses literals with too many digits:
        - swapTokensAtAmount = 1000000000 * (10 ** 18) (Fat.sol#1360)
FAT.slitherConstructorVariables() (Fat.sol#1344-2019) uses literals with too many digits:
        - _maxWalletToken = 2000000000 * (10 ** 18) (Fat.sol#1361)
FAT.slitherConstructorVariables() (Fat.sol#1344-2019) uses literals with too many digits:
        - gasForProcessing = 300000 (Fat.sol#1404)
FATDividendTracker.constructor() (Fat.sol#2041-2044) uses literals with too many digits:
        - minimumTokenBalanceForDividends = 100000000 * (10 ** 18) (Fat.sol#2043)
FATDividendTracker.getAccountAtIndex(uint256) (Fat.sol#2138-2155) uses literals with too many digits:
        - (0x0000000000000000000000000000000000000000,- 1,- 1,0,0,0,0,0) (Fat.sol#2149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

DividendPayingToken.lastAmount (Fat.sol#1110) is never used in FATDividendTracker (Fat.sol#2021-2241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DividendPayingToken.lastAmount (Fat.sol#1110) should be constant
FAT.DividendToken (Fat.sol#1350) should be constant
FAT.maxBuyTranscationAmount (Fat.sol#1358) should be constant
FAT.maxSellTransactionAmount (Fat.sol#1359) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

get(IterableMapping.Map,address) should be declared external:
        - IterableMapping.get(IterableMapping.Map,address) (Fat.sol#329-331)
getIndexOfKey(IterableMapping.Map,address) should be declared external:
        - IterableMapping.getIndexOfKey(IterableMapping.Map,address) (Fat.sol#333-338)
getKeyAtIndex(IterableMapping.Map,uint256) should be declared external:
        - IterableMapping.getKeyAtIndex(IterableMapping.Map,uint256) (Fat.sol#340-342)
size(IterableMapping.Map) should be declared external:
        - IterableMapping.size(IterableMapping.Map) (Fat.sol#344-346)
```

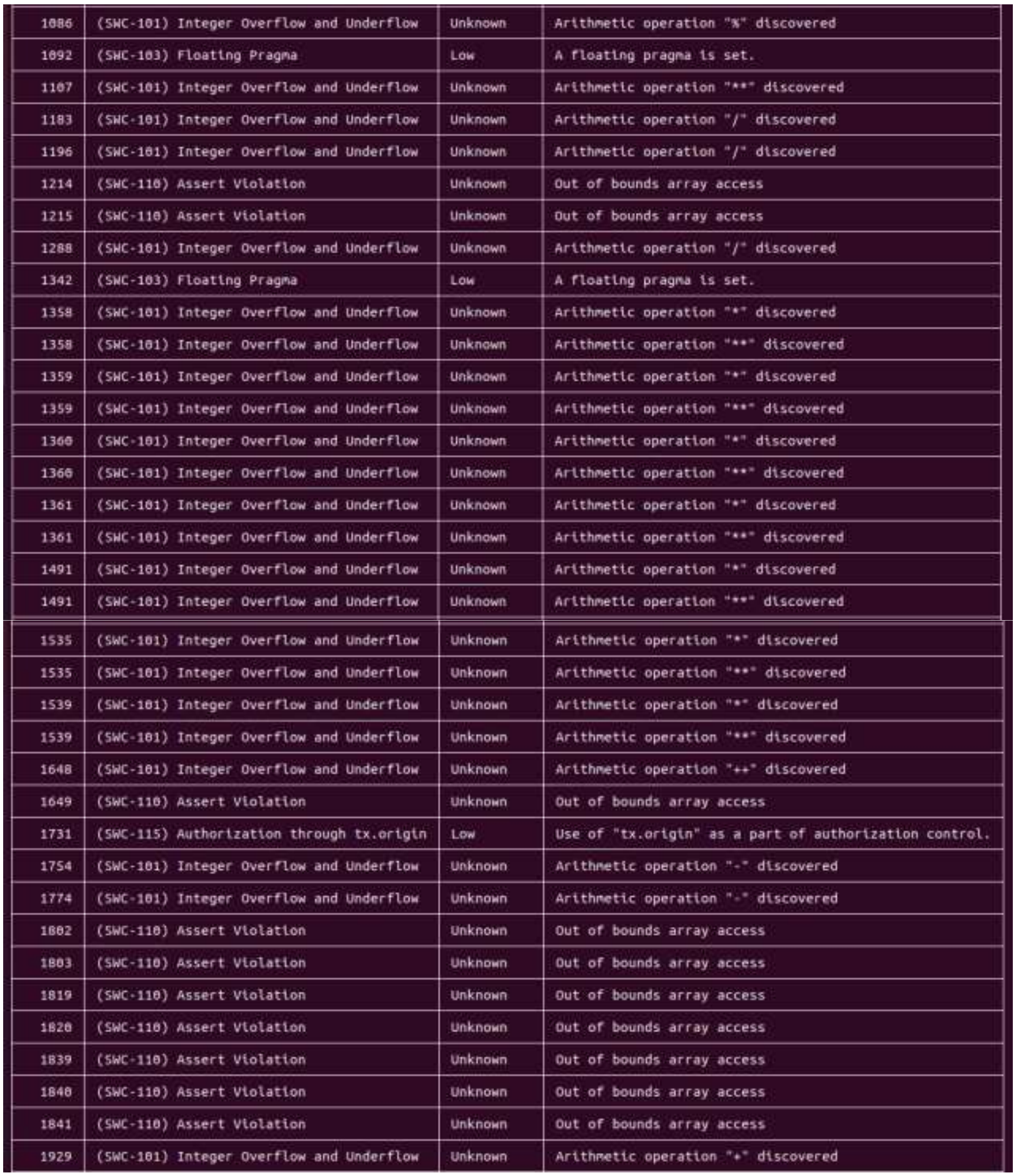


```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Fat.sol#430-433)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Fat.sol#439-443)
name() should be declared external:
        - ERC20.name() (Fat.sol#632-634)
symbol() should be declared external:
        - ERC20.symbol() (Fat.sol#640-642)
decimals() should be declared external:
        - ERC20.decimals() (Fat.sol#657-659)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Fat.sol#683-686)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (Fat.sol#691-693)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Fat.sol#702-705)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Fat.sol#720-724)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Fat.sol#738-741)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Fat.sol#757-760)
updateMasterContract(address) should be declared external:
        - DividendPayingToken.updateMasterContract(address) (Fat.sol#1149-1151)
updateUniswapV2Router(address) should be declared external:
        - DividendPayingToken.updateUniswapV2Router(address) (Fat.sol#1153-1155)
updateDividendTokenUser(address,address) should be declared external:
        - DividendPayingToken.updateDividendTokenUser(address,address) (Fat.sol#1157-1159)
getDividendTokenUser(address) should be declared external:
        - DividendPayingToken.getDividendTokenUser(address) (Fat.sol#1161-1163)
distributeDividends() should be declared external:
        - DividendPayingToken.distributeDividends() (Fat.sol#1170-1189)
distributeTokenDividends(uint256) should be declared external:
        - DividendPayingToken.distributeTokenDividends(uint256) (Fat.sol#1191-1202)
withdrawDividend() should be declared external:
        - DividendPayingToken.withdrawDividend() (Fat.sol#1206-1208)
        - FATDividendTracker.withdrawDividend() (Fat.sol#2058-2060)
dividendOf(address) should be declared external:
        - DividendPayingToken.dividendOf(address) (Fat.sol#1264-1266)
withdrawnDividendOf(address) should be declared external:
        - DividendPayingToken.withdrawnDividendOf(address) (Fat.sol#1278-1280)
blacklistAddress(address,bool) should be declared external:
        - FAT.blacklistAddress(address,bool) (Fat.sol#1628-1633)
excludeFromDividends(address) should be declared external:
        - FAT.excludeFromDividends(address) (Fat.sol#1635-1639)
enableDividends(address) should be declared external:
        - FAT.enableDividends(address) (Fat.sol#1641-1645)
updateGasForProcessing(uint256) should be declared external:
        - FAT.updateGasForProcessing(uint256) (Fat.sol#1672-1677)
isExcludedFromFees(address) should be declared external:
        - FAT.isExcludedFromFees(address) (Fat.sol#1691-1693)
withdrawableDividendOf(address) should be declared external:
        - FAT.withdrawableDividendOf(address) (Fat.sol#1695-1697)
dividendTokenBalanceOf(address) should be declared external:
        - FAT.dividendTokenBalanceOf(address) (Fat.sol#1699-1701)
getAccountAtIndex(uint256) should be declared external:
        - FATDividendTracker.getAccountAtIndex(uint256) (Fat.sol#2138-2155)
process(uint256) should be declared external:
        - FATDividendTracker.process(uint256) (Fat.sol#2182-2227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

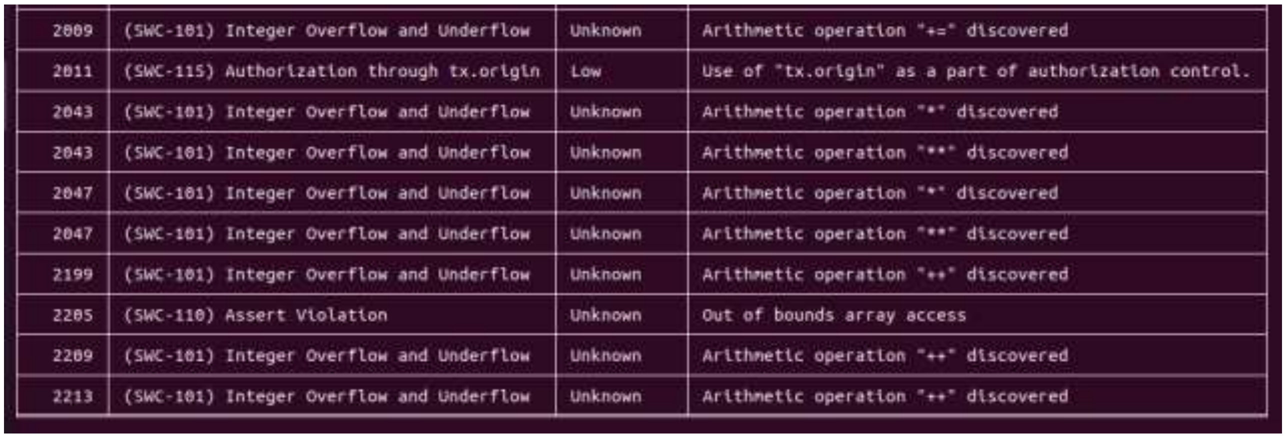
MythX: -



```
Report for Fat.sol
https://dashboard.mythx.io/#/console/analyses/59fc8bca-c555-433b-afb9-fe95edbd0787

│ Line │ SWC Title                                   │ Severity │ Short Description                            │
│    1 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│   78 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  103 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  200 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  245 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  264 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  318 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  341 │ (SWC-110) Assert Violation                  │ Unknown  │ Out of bounds array access                   │
│  368 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Compiler-rewritable "<uint> - 1" discovered  │
│  368 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│  369 │ (SWC-110) Assert Violation                  │ Unknown  │ Out of bounds array access                   │
│  374 │ (SWC-110) Assert Violation                  │ Unknown  │ Out of bounds array access                   │
│  380 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  447 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  472 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  513 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  525 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "**" discovered         │
│  527 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "*" discovered          │
│  528 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│  535 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "**" discovered         │
│  537 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│  541 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│  543 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│  547 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "+" discovered          │
│  559 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  575 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  877 │ (SWC-103) Floating Pragma                   │ Low      │ A floating pragma is set.                    │
│  899 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "+" discovered          │
│  911 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│  924 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "*" discovered          │
│  925 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│  936 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│  946 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "%" discovered          │
│  960 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "+" discovered          │
│  977 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│  992 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "*" discovered          │
│  993 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│ 1011 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
│ 1028 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "%" discovered          │
│ 1046 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "-" discovered          │
│ 1066 │ (SWC-101) Integer Overflow and Underflow    │ Unknown  │ Arithmetic operation "/" discovered          │
```

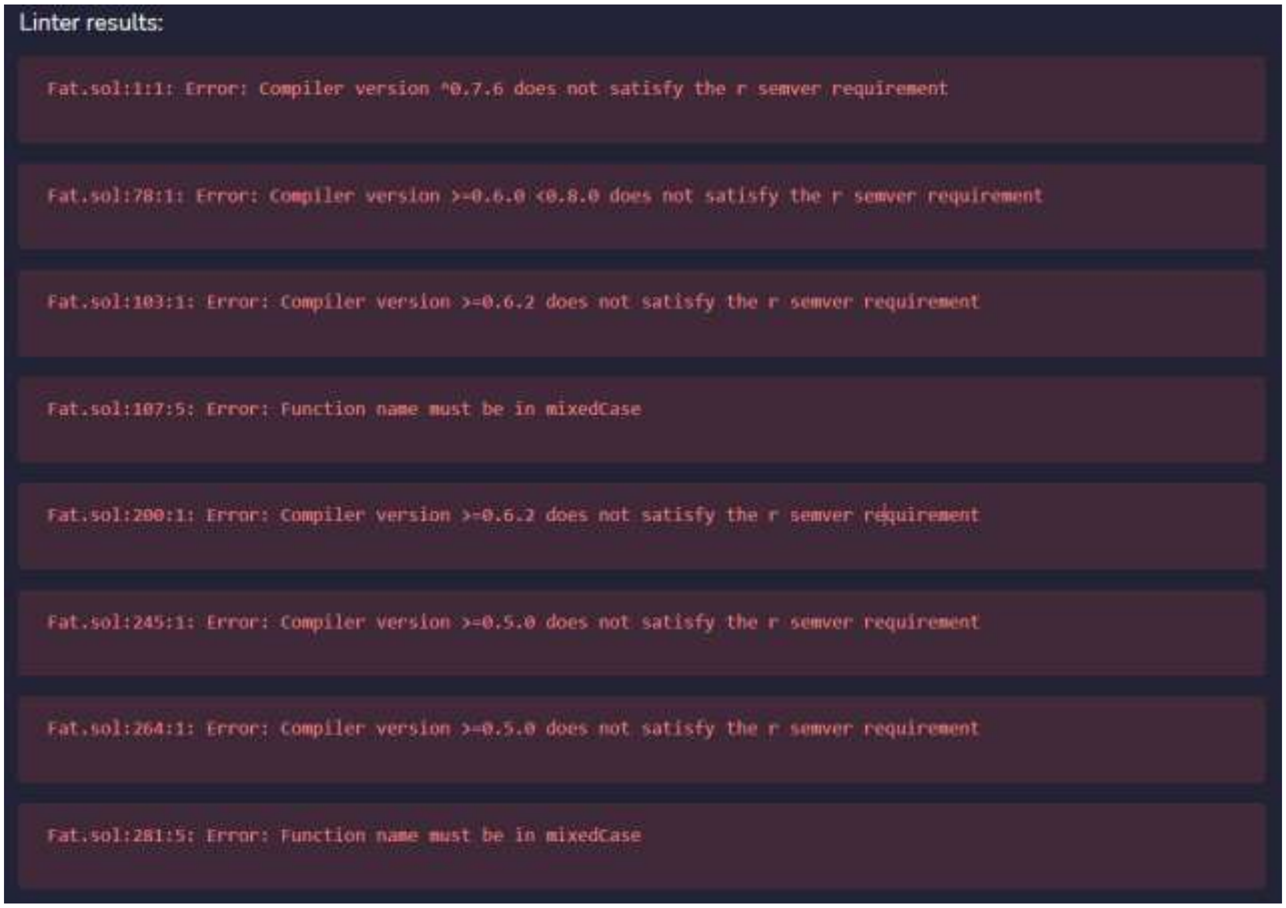


1086	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
1092	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1107	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1183	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
1196	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
1214	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1215	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1288	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
1342	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1358	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1358	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1359	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1359	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1360	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1360	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1361	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1361	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1491	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1491	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1535	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1535	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1539	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1539	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1648	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
1649	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1731	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.
1754	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
1774	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
1802	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1803	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1819	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1820	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1839	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1840	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1841	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1929	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered



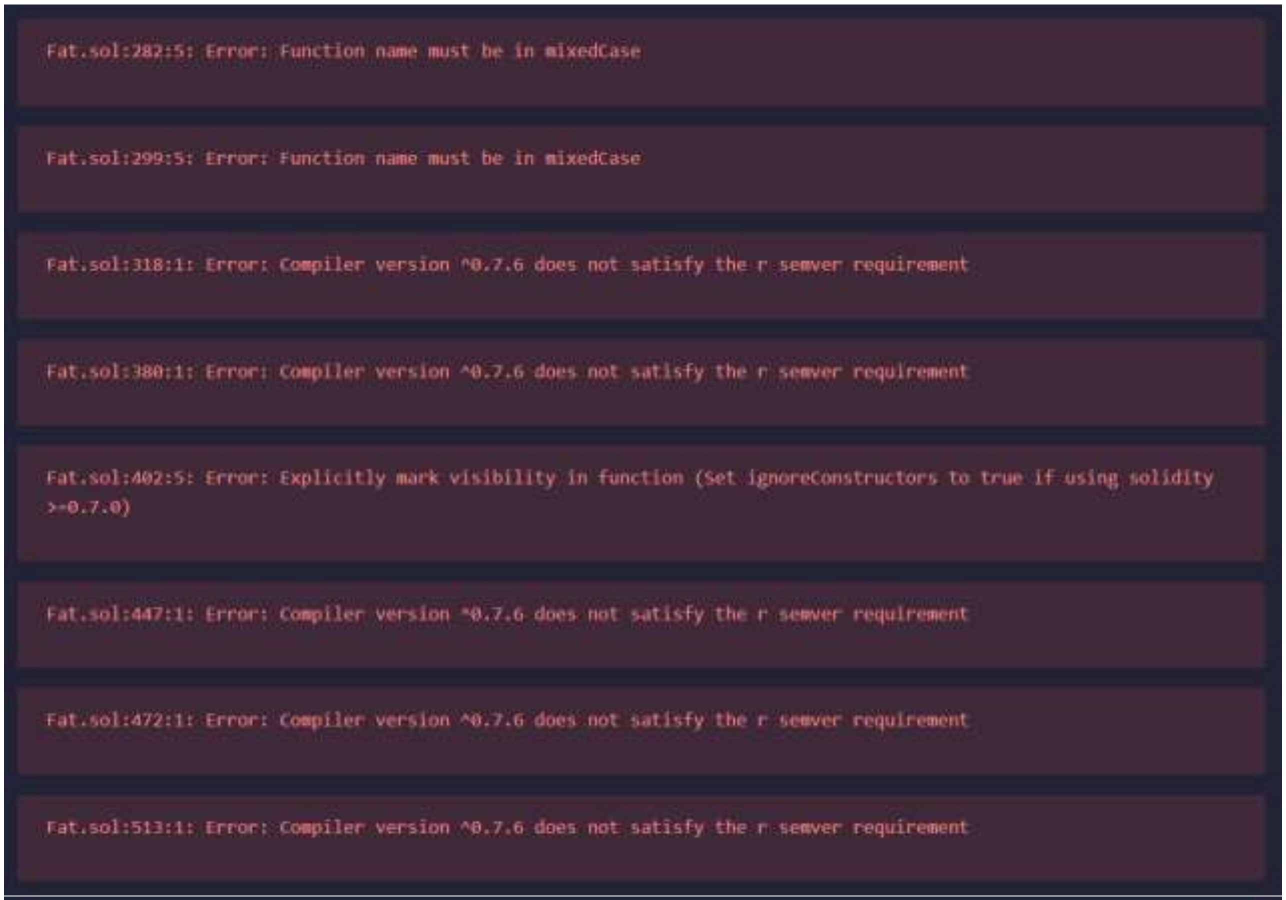
```
| 2009 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered                        |
| 2011 | (SWC-115) Authorization through tx.origin | Low     | Use of "tx.origin" as a part of authorization control.     |
| 2043 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered                         |
| 2043 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "**" discovered                        |
| 2047 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered                         |
| 2047 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "**" discovered                        |
| 2199 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered                        |
| 2205 | (SWC-110) Assert Violation               | Unknown | Out of bounds array access                                  |
| 2209 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered                        |
| 2213 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered                        |
```

Solhint: -

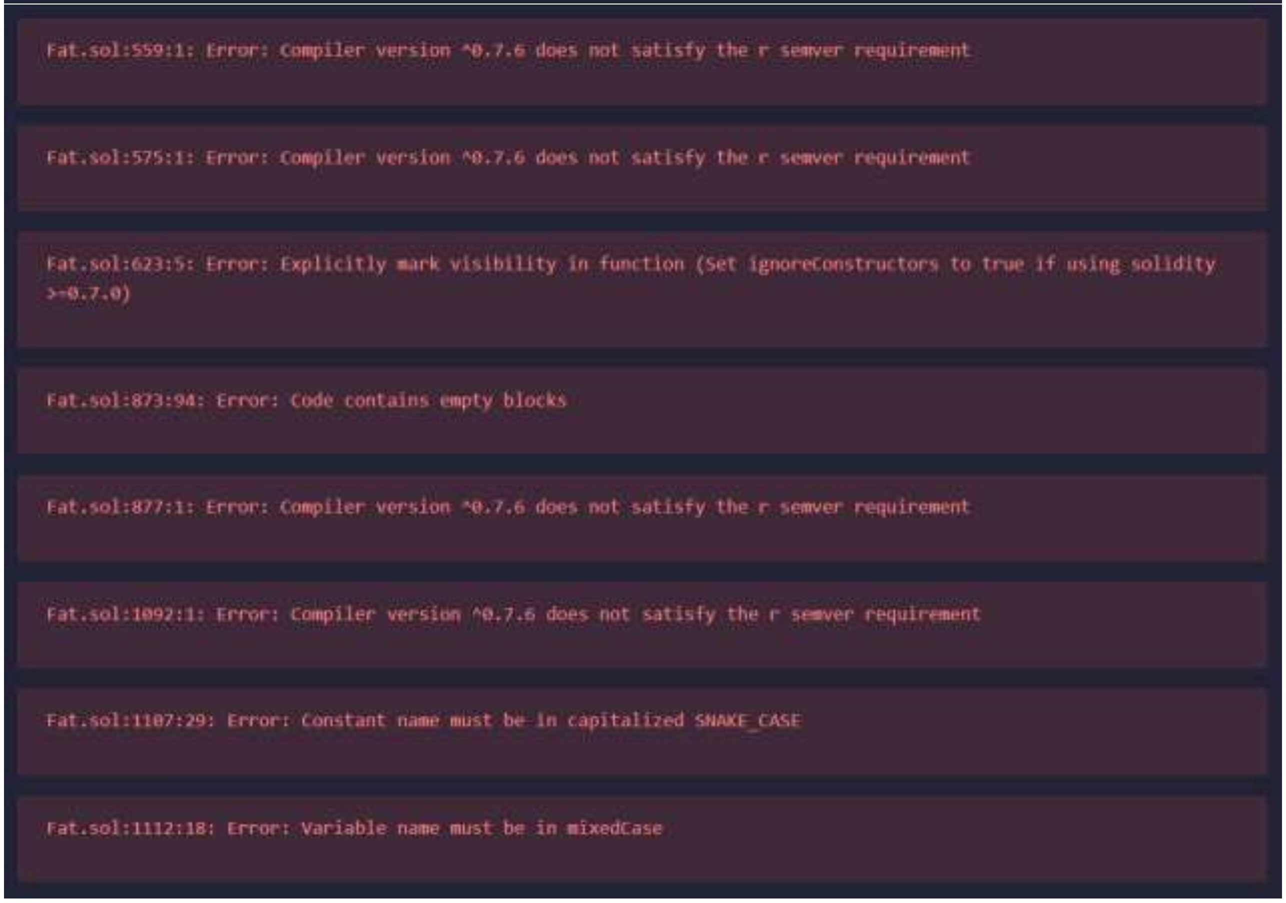


Linter results:

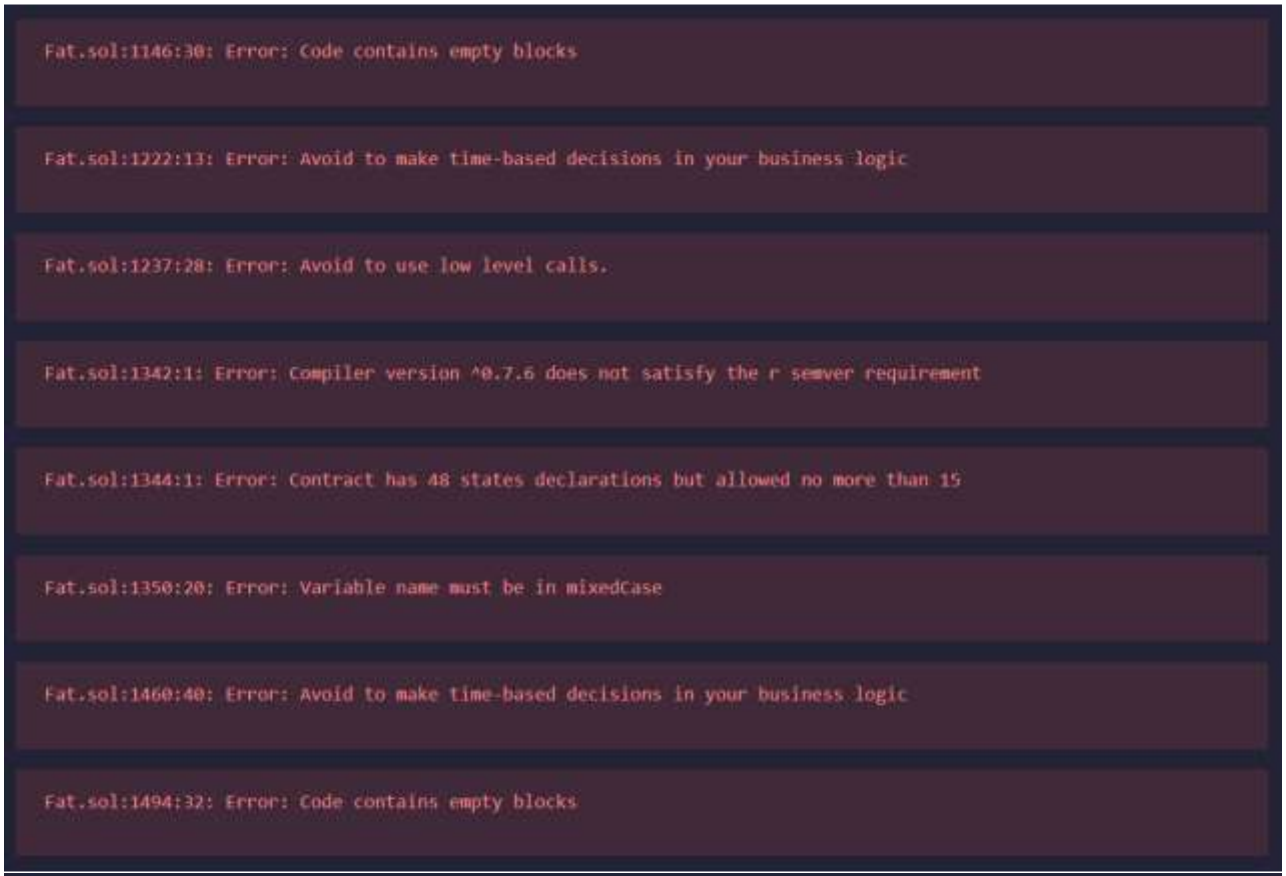
```
Fat.sol:1:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:78:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy the r semver requirement

Fat.sol:103:1: Error: Compiler version >=0.6.2 does not satisfy the r semver requirement

Fat.sol:107:5: Error: Function name must be in mixedCase

Fat.sol:200:1: Error: Compiler version >=0.6.2 does not satisfy the r semver requirement

Fat.sol:245:1: Error: Compiler version >=0.5.0 does not satisfy the r semver requirement

Fat.sol:264:1: Error: Compiler version >=0.5.0 does not satisfy the r semver requirement

Fat.sol:281:5: Error: Function name must be in mixedCase
```



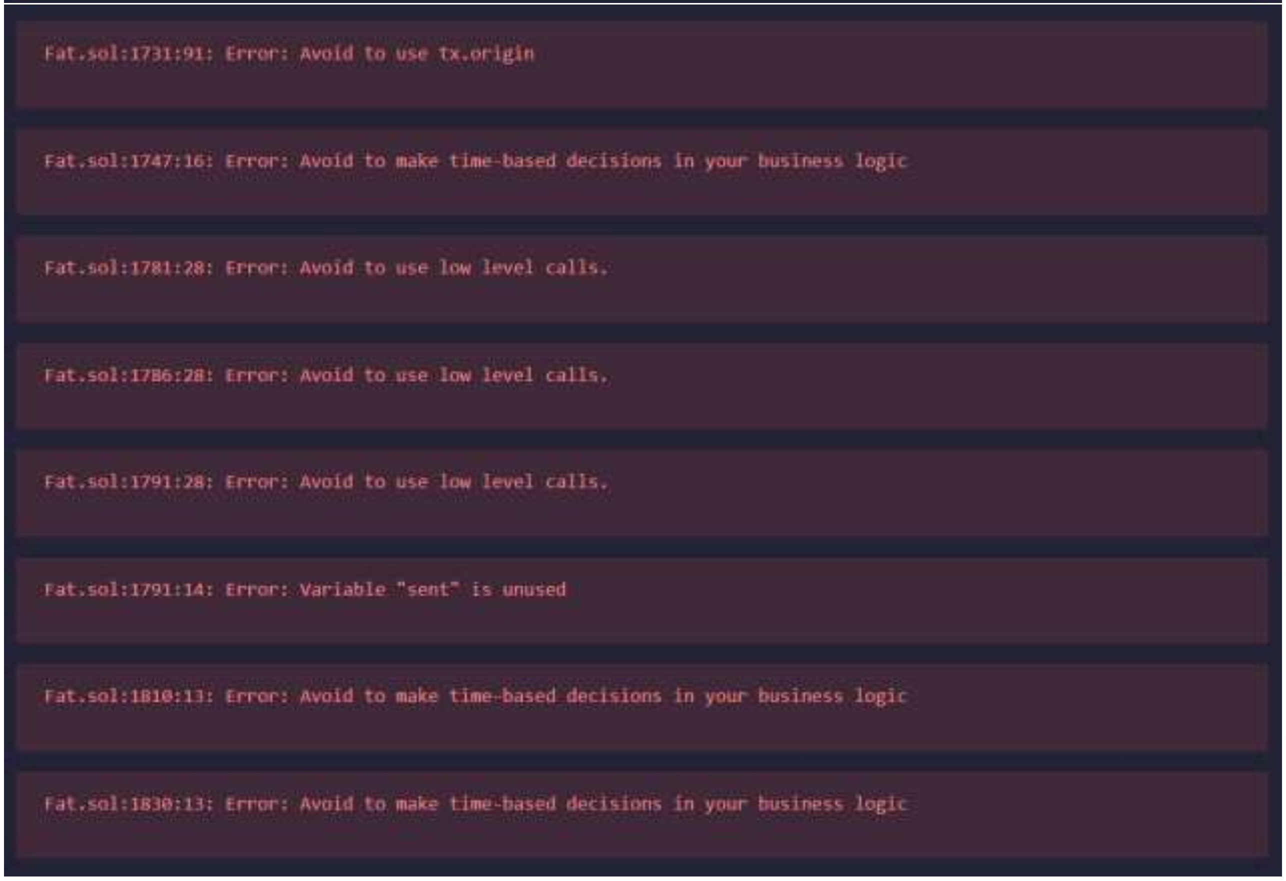
```
Fat.sol:282:5: Error: Function name must be in mixedCase

Fat.sol:299:5: Error: Function name must be in mixedCase

Fat.sol:318:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:380:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:402:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity
>=0.7.0)

Fat.sol:447:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:472:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:513:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement
```



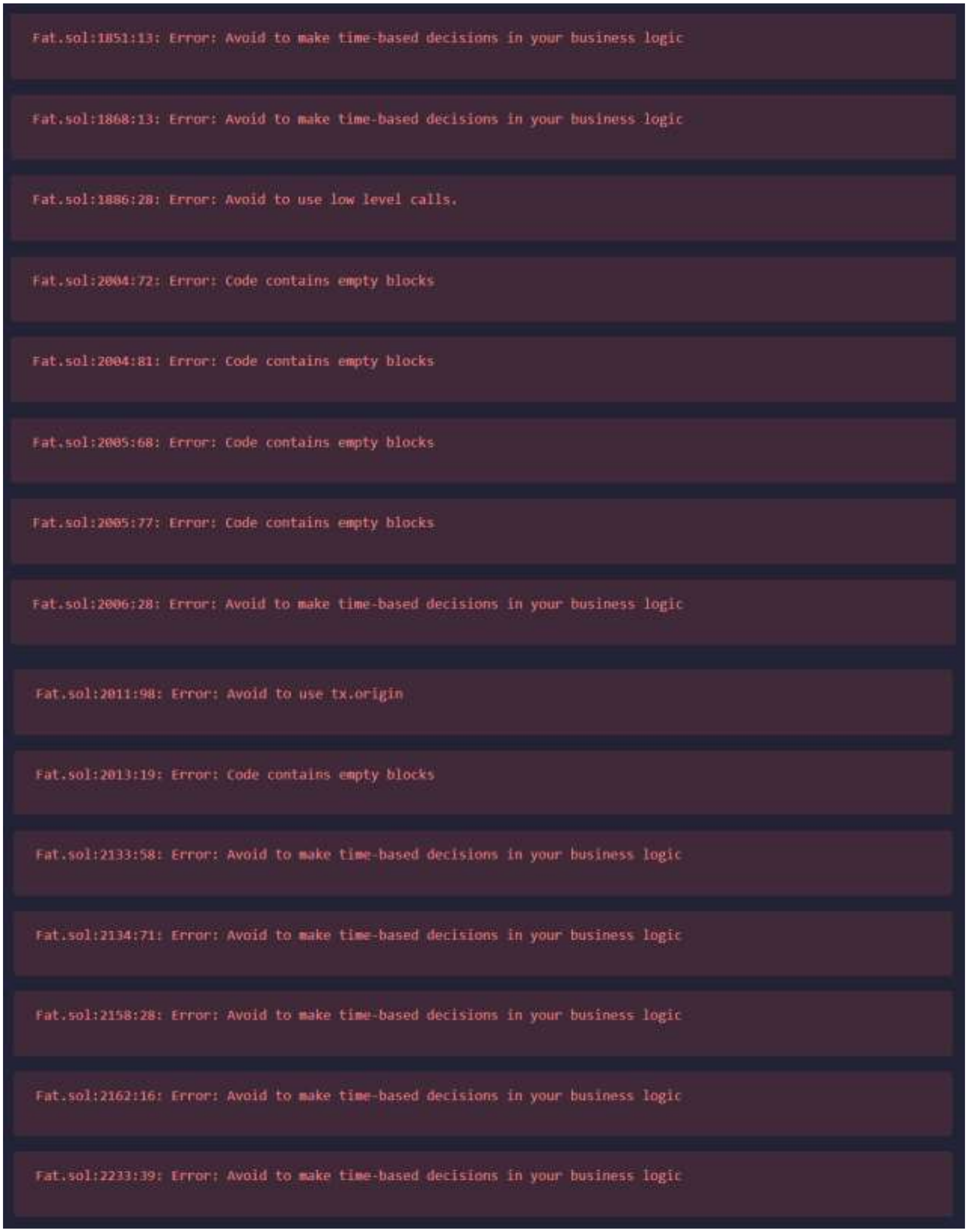
```
Fat.sol:559:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:575:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:623:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity
>=0.7.0)

Fat.sol:873:94: Error: Code contains empty blocks

Fat.sol:877:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:1092:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:1107:29: Error: Constant name must be in capitalized SNAKE_CASE

Fat.sol:1112:18: Error: Variable name must be in mixedCase
```



```
Fat.sol:1146:30: Error: Code contains empty blocks

Fat.sol:1222:13: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1237:28: Error: Avoid to use low level calls.

Fat.sol:1342:1: Error: Compiler version ^0.7.6 does not satisfy the r semver requirement

Fat.sol:1344:1: Error: Contract has 48 states declarations but allowed no more than 15

Fat.sol:1350:20: Error: Variable name must be in mixedCase

Fat.sol:1460:40: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1494:32: Error: Code contains empty blocks
```



```
Fat.sol:1731:91: Error: Avoid to use tx.origin

Fat.sol:1747:16: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1781:28: Error: Avoid to use low level calls.

Fat.sol:1786:28: Error: Avoid to use low level calls.

Fat.sol:1791:28: Error: Avoid to use low level calls.

Fat.sol:1791:14: Error: Variable "sent" is unused

Fat.sol:1810:13: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1830:13: Error: Avoid to make time-based decisions in your business logic
```



```
Fat.sol:1851:13: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1868:13: Error: Avoid to make time-based decisions in your business logic

Fat.sol:1886:28: Error: Avoid to use low level calls.

Fat.sol:2004:72: Error: Code contains empty blocks

Fat.sol:2004:81: Error: Code contains empty blocks

Fat.sol:2005:68: Error: Code contains empty blocks

Fat.sol:2005:77: Error: Code contains empty blocks

Fat.sol:2006:28: Error: Avoid to make time-based decisions in your business logic

Fat.sol:2011:98: Error: Avoid to use tx.origin

Fat.sol:2013:19: Error: Code contains empty blocks

Fat.sol:2133:58: Error: Avoid to make time-based decisions in your business logic

Fat.sol:2134:71: Error: Avoid to make time-based decisions in your business logic

Fat.sol:2158:28: Error: Avoid to make time-based decisions in your business logic

Fat.sol:2162:16: Error: Avoid to make time-based decisions in your business logic

Fat.sol:2233:39: Error: Avoid to make time-based decisions in your business logic
```

Basic Coding Bugs

1. **Constructor Mismatch**
   - Description: Whether the contract name and its constructor are not identical to each other.
   - Result: PASSED
   - Severity: Critical

2. **Ownership Takeover**
   - Description: Whether the set owner function is not protected.
   - Result: PASSED
   - Severity: Critical

3. **Redundant Fallback Function**
   - Description: Whether the contract has a redundant fallback function.
   - Result: PASSED
   - Severity: Critical

4. **Overflows & Underflows**
   - Description: Whether the contract has general overflow or underflow vulnerabilities
   - Result: Found
   - Severity: Critical (In this audit High Impact found)

5. **Reentrancy**
   - Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.
   - Result: Found
   - Severity: Critical (In this audit Low Impact found)

6. **MONEY-Giving Bug**
   - Description: Whether the contract returns funds to an arbitrary address.
   - Result: PASSED
   - Severity: High

7. **Blackhole**
    - Description: Whether the contract locks ETH indefinitely: merely in without out.
    - Result: PASSED
    - Severity: High

8. **Unauthorized Self-Destruct**
    - Description: Whether the contract can be killed by any arbitrary address.
    - Result: PASSED
    - Severity: Medium

9. **Revert DoS**
    - Description: Whether the contract is vulnerable to DoS attack because of unexpected revert.
    - Result: PASSED
    - Severity: Medium

10. **Unchecked External Call**
    - Description: Whether the contract has any external call without checking the return value.
    - Result: PASSED
    - Severity: Medium

11. **Gasless Send**
    - Description: Whether the contract is vulnerable to gasless send.
    - Result: PASSED
    - Severity: Medium

12. **Send Instead of Transfer**
    - Description: Whether the contract uses send instead of transfer.
    - Result: PASSED
    - Severity: Medium

13. **Costly Loop**

- Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.
- Result: PASSED
- Severity: Medium

14. **(Unsafe) Use of Untrusted Libraries**

- Description: Whether the contract use any suspicious libraries.
- Result: PASSED
- Severity: Medium

15. **(Unsafe) Use of Predictable Variables**

- Description: Whether the contract contains any randomness variable, but its value can be predicated.
- Result: PASSED
- Severity: Medium

16. **Transaction Ordering Dependence**

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: PASSED
- Severity: Medium

17. **Deprecated Uses**

- Description: Whether the contract use the deprecated tx.origin to perform the authorization.
- Result: Found(In this audit Medium Impact found)
- Severity: Medium

Semantic Consistency Checks

- Description: Whether the semantic of the white paper is different from the implementation of the contract.
- Result: PASSED
- Severity: Critical

Conclusion

In this audit, we thoroughly analyzed FATBULL's Fat Smart Contract. The current code base is well organized but there are promptly some high and low-level issues along with the compilation issues found, in the first phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

About eNebula Solutions

We believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The eNebula Solutions team has skills for reviewing code in C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, and JavaScript for common security vulnerabilities & specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent and open about the work we do.

For more information about our security consulting,
please mail us at – contact@enebula.in