QuillAudits

Audit Report September, 2021

For

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Yummy Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	2	0
Closed	0	0	0	0

Introduction

During the period of **September 8, 2021, to September 13, 2021** - QuillAudits Team performed a security audit for YummyToken smart contract.

The code for the audit was taken from the following official File:
https://bscscan.com/address/0xb003c68917bab76812797d1b8056822f48e2e4fe#code

Issues Found

High severity issues

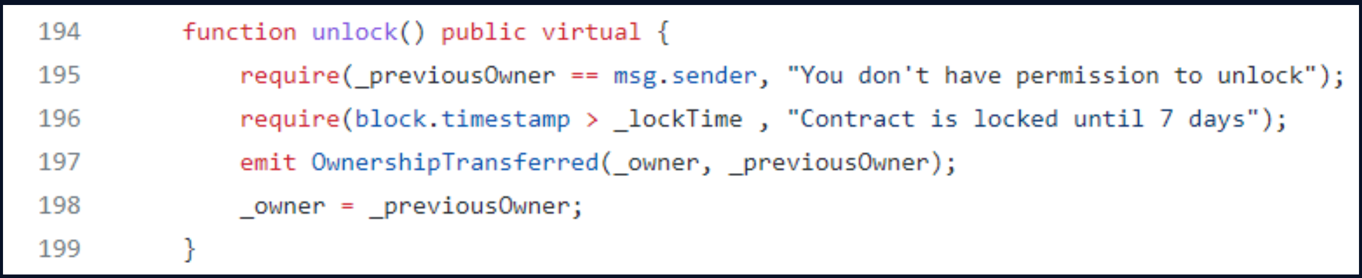
No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Block values as a proxy for time**



```
    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have permission to unlock");
        require(block.timestamp > _lockTime , "Contract is locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
```

Description

Here in function unlock() A control flow decision is made based on The 'block.timestamp' environment variable.

Note that the values of variables like coinbase, gaslimit, block number, and timestamp are predictable and can be manipulated by malicious miners. 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 the use of these variables introduces a certain level of trust into miners.

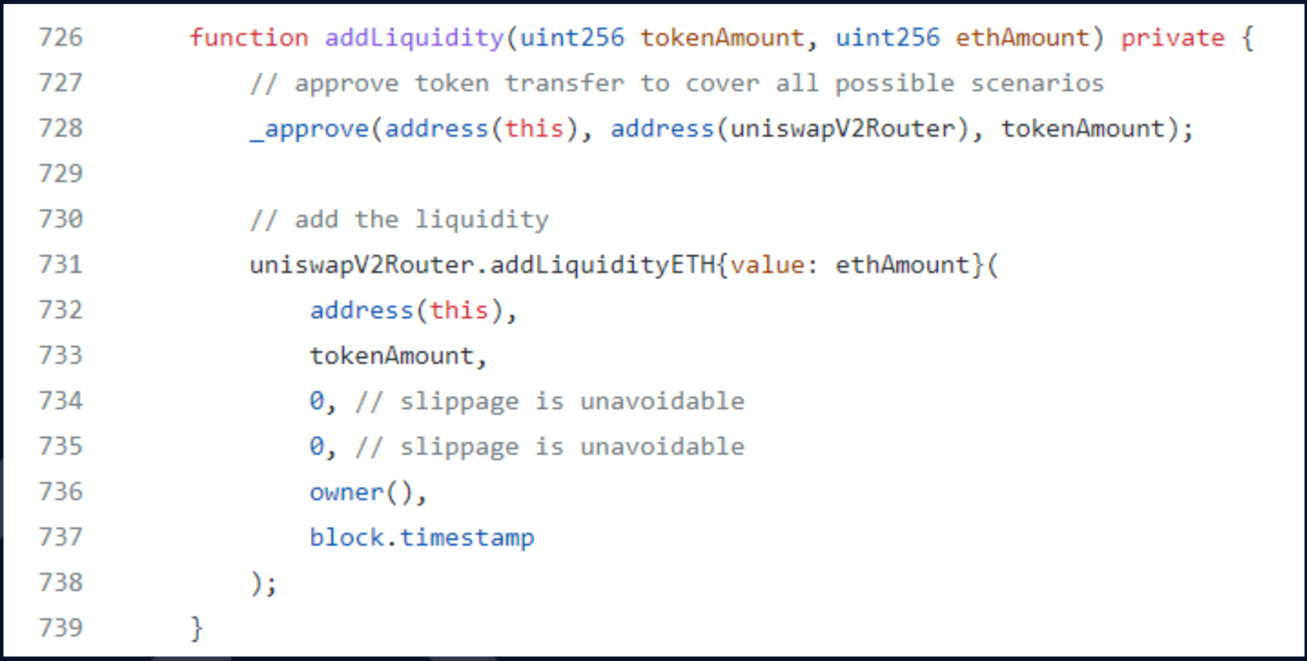
Remediation

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 of oracles.

Reference
https://swcregistry.io/docs/SWC-116
Secure Development Recommendations - Ethereum Smart Contract Best Practices

Status: **Acknowledged by the Auditee**
The Auditee responded that the unlock function does not require a precise timestamp. If a miner manipulates the timestamp by a couple of blocks the contract would just unlock slightly earlier or later. Neither of those outcomes is critical.
And also, they've confirmed that The block timestamp is not being used as a source of randomness in here.

2. Weak Sources of Randomness from Chain Attributes



```
    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
    }
```

Description
Here in function addLiquidity() ' block.timestamp ' is used as a source of randomness, unless you know what you are doing.

Both 'block.timestamp', 'now' or 'blockhash' can be influenced by miners to some extent.
For example, the use of a block.timestamp is insecure, as a miner can choose to provide any timestamp within a few seconds and still get his block accepted by others. Use of blockhash, block.difficulty and other fields are also insecure, as they're controlled by the miner.

Remediation

1. Using a commitment scheme, e.g. RANDAO.
2. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in an oracle, thus it may be reasonable to use multiple oracles.
3. Using Bitcoin block hashes, as they are more expensive to mine.

Reference

SWC-120 · Overview (swcregistry.io)
https://docs.soliditylang.org/en/v0.6.12/units-and-global-variables.html?highlight=block.timestamp#block-and-transaction-properties

Status: **Acknowledged by the Auditee**

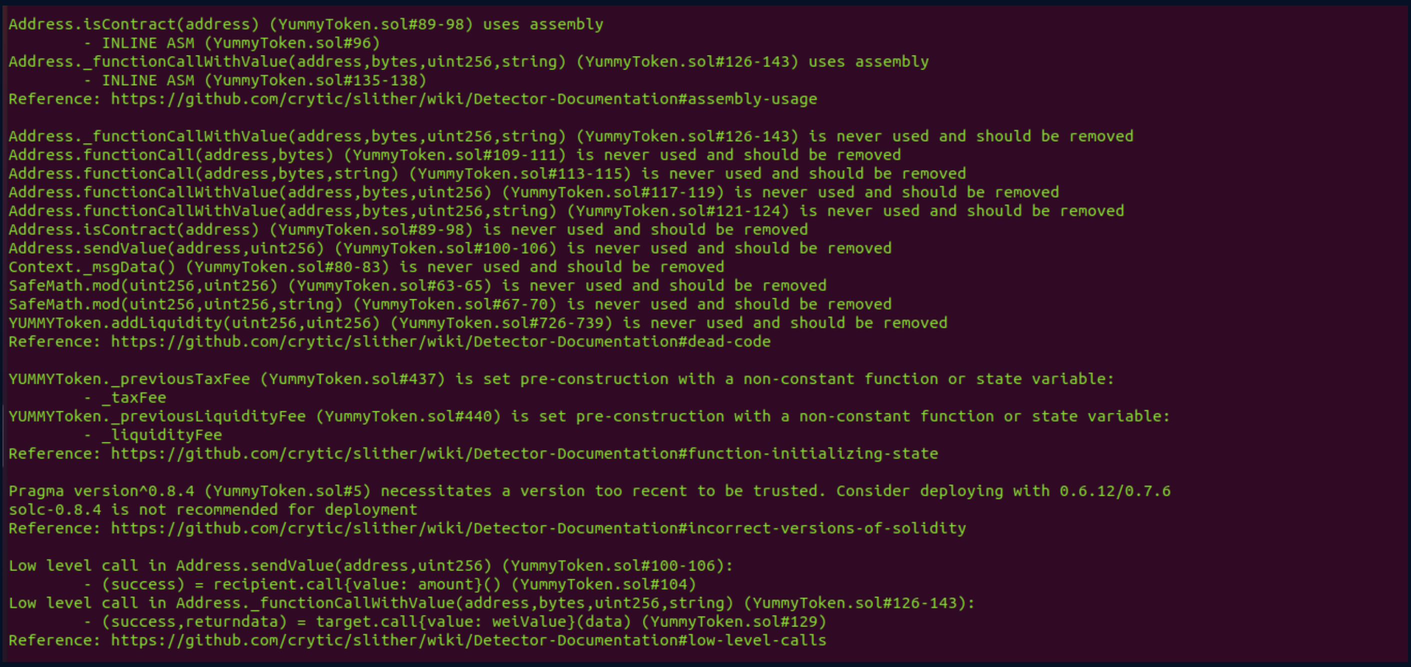
The auditee confirms the block.timestamp here is being used to set the deadline for the liquidity add transaction. It is not being used as a source of randomness.

Functional Tests

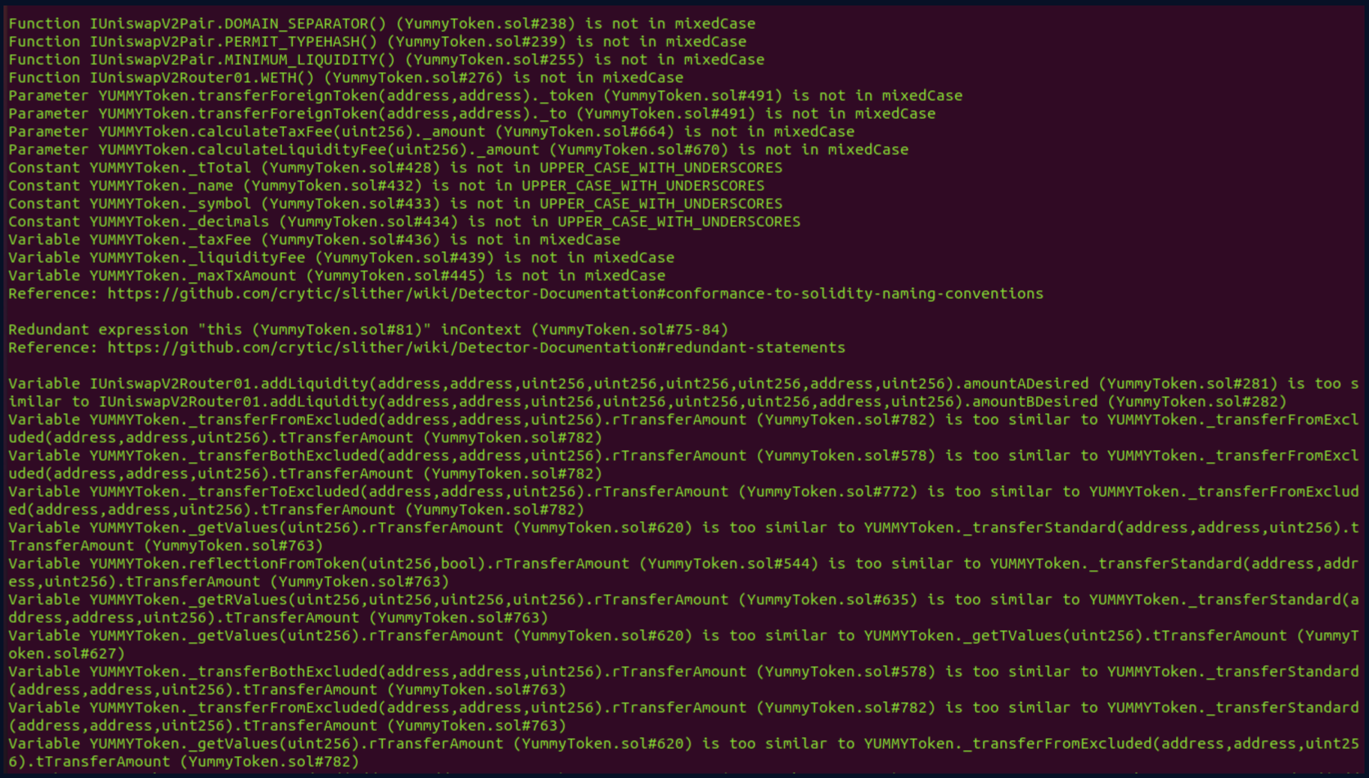
Function Names	Testing results
approve	Passed
deliver	Passed
excludeFromReward	Passed
includeInReward	Passed
lock	Passed
transfer	Passed
transferForeignToken	Passed
transferFrom	Passed
reflectionFromToken	Passed
tokenFromReflection	Passed

Automated Testing

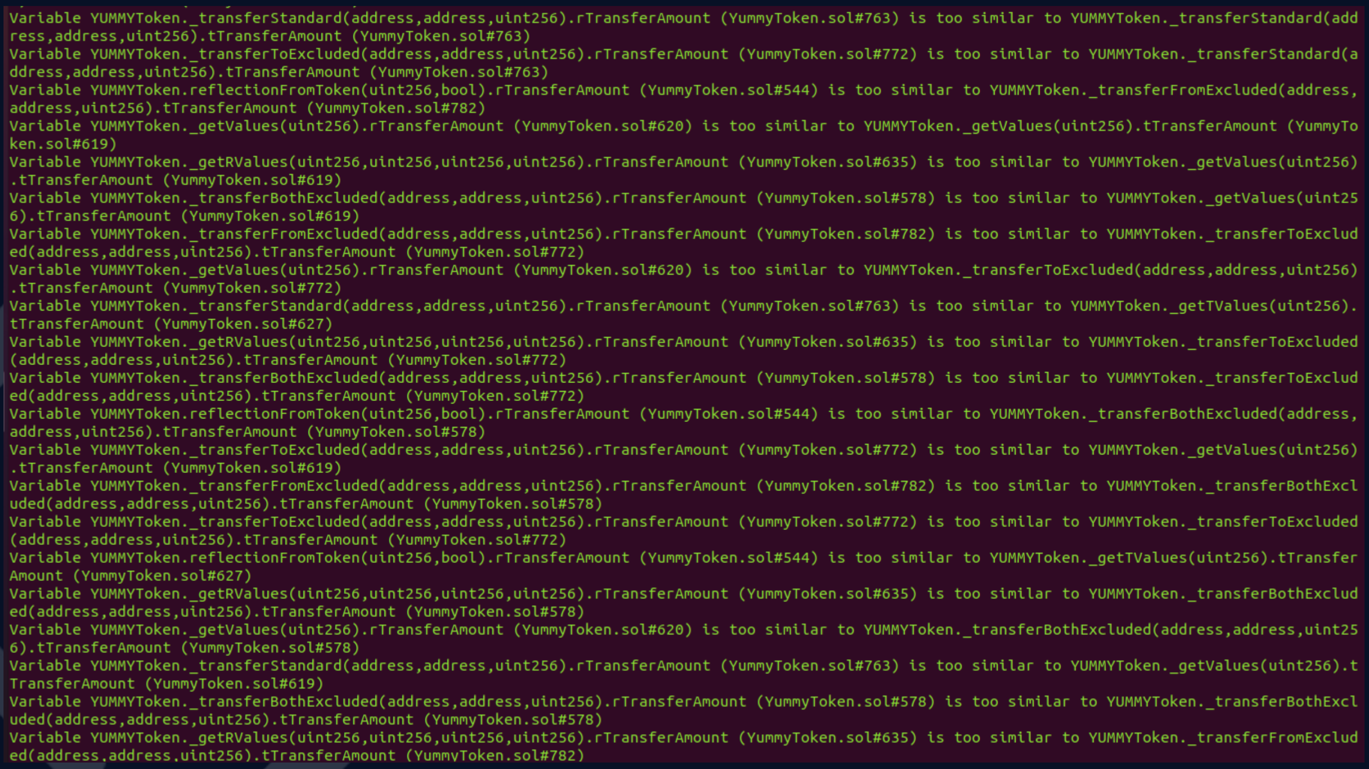
Slither



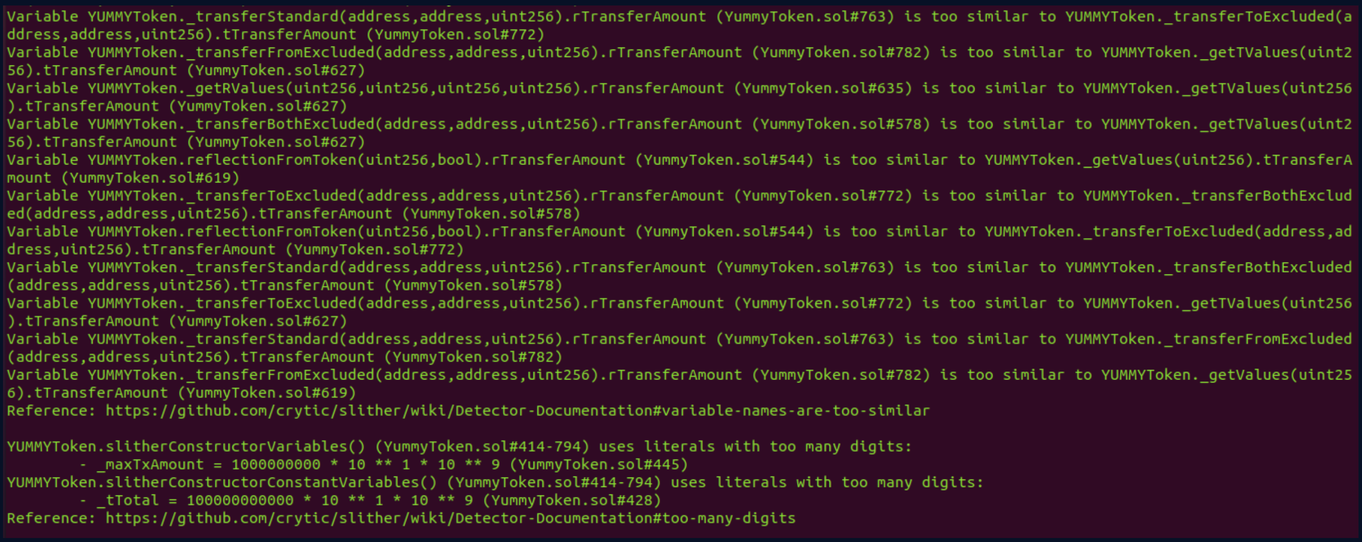
```
Address.isContract(address) (YummyToken.sol#89-98) uses assembly
        - INLINE ASM (YummyToken.sol#96)
Address._functionCallWithValue(address,bytes,uint256,string) (YummyToken.sol#126-143) uses assembly
        - INLINE ASM (YummyToken.sol#135-138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (YummyToken.sol#126-143) is never used and should be removed
Address.functionCall(address,bytes) (YummyToken.sol#109-111) is never used and should be removed
Address.functionCall(address,bytes,string) (YummyToken.sol#113-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (YummyToken.sol#117-119) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (YummyToken.sol#121-124) is never used and should be removed
Address.isContract(address) (YummyToken.sol#89-98) is never used and should be removed
Address.sendValue(address,uint256) (YummyToken.sol#100-106) is never used and should be removed
Context._msgData() (YummyToken.sol#80-83) is never used and should be removed
SafeMath.mod(uint256,uint256) (YummyToken.sol#63-65) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (YummyToken.sol#67-70) is never used and should be removed
YUMMYToken.addLiquidity(uint256,uint256) (YummyToken.sol#726-739) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

YUMMYToken._previousTaxFee (YummyToken.sol#437) is set pre-construction with a non-constant function or state variable:
        - _taxFee
YUMMYToken._previousLiquidityFee (YummyToken.sol#440) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.4 (YummyToken.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (YummyToken.sol#100-106):
        - (success) = recipient.call{value: amount}() (YummyToken.sol#104)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (YummyToken.sol#126-143):
        - (success,returndata) = target.call{value: weiValue}(data) (YummyToken.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



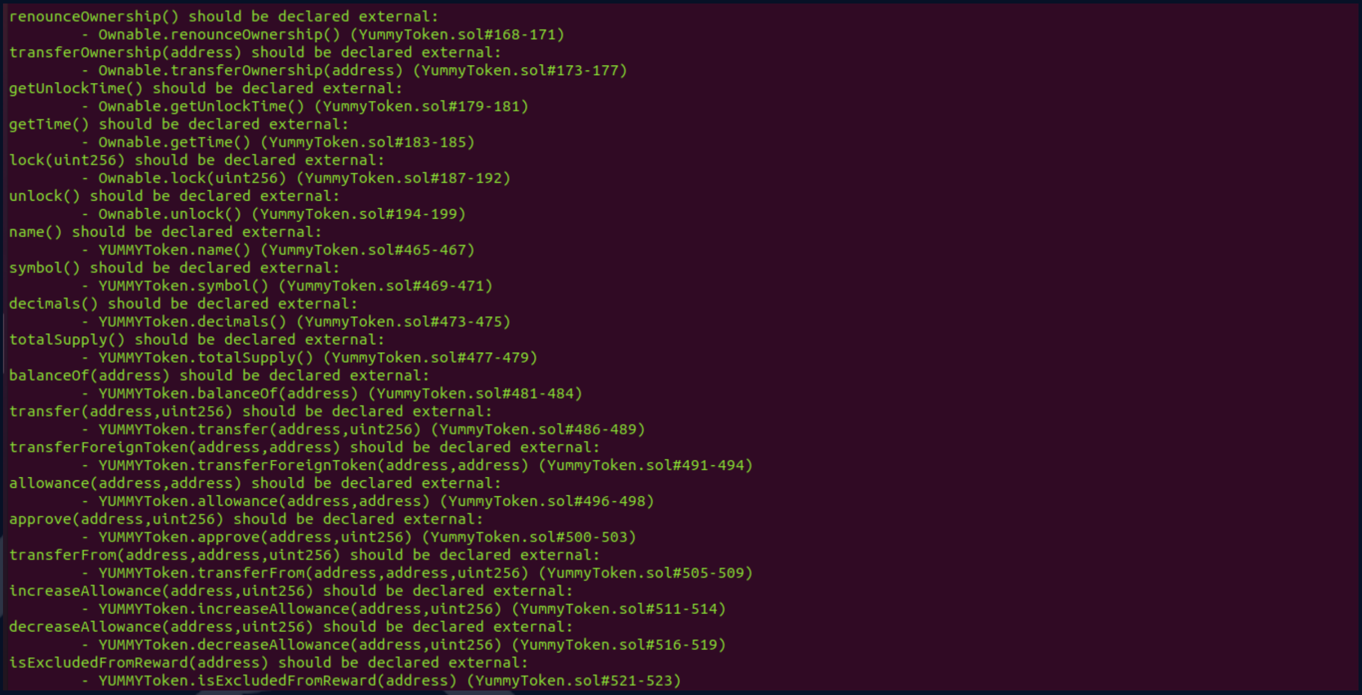
```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (YummyToken.sol#238) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (YummyToken.sol#239) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (YummyToken.sol#255) is not in mixedCase
Function IUniswapV2Router01.WETH() (YummyToken.sol#276) is not in mixedCase
Parameter YUMMYToken.transferForeignToken(address,address)._token (YummyToken.sol#491) is not in mixedCase
Parameter YUMMYToken.transferForeignToken(address,address)._to (YummyToken.sol#491) is not in mixedCase
Parameter YUMMYToken.calculateTaxFee(uint256)._amount (YummyToken.sol#664) is not in mixedCase
Parameter YUMMYToken.calculateLiquidityFee(uint256)._amount (YummyToken.sol#670) is not in mixedCase
Constant YUMMYToken._tTotal (YummyToken.sol#428) is not in UPPER_CASE_WITH_UNDERSCORES
Constant YUMMYToken._name (YummyToken.sol#432) is not in UPPER_CASE_WITH_UNDERSCORES
Constant YUMMYToken._symbol (YummyToken.sol#433) is not in UPPER_CASE_WITH_UNDERSCORES
Constant YUMMYToken._decimals (YummyToken.sol#434) is not in UPPER_CASE_WITH_UNDERSCORES
Variable YUMMYToken._taxFee (YummyToken.sol#436) is not in mixedCase
Variable YUMMYToken._liquidityFee (YummyToken.sol#439) is not in mixedCase
Variable YUMMYToken._maxTxAmount (YummyToken.sol#445) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (YummyToken.sol#81)" inContext (YummyToken.sol#75-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (YummyToken.sol#281) is too s
imilar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (YummyToken.sol#282)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._transferFromExcl
uded(address,address,uint256).tTransferAmount (YummyToken.sol#782)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._transferFromExcl
uded(address,address,uint256).tTransferAmount (YummyToken.sol#782)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._transferFromExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#782)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._transferStandard(address,address,uint256).t
TransferAmount (YummyToken.sol#763)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._transferStandard(address,addr
ess,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._transferStandard(a
ddress,address,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._getTValues(uint256).tTransferAmount (YummyT
oken.sol#627)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._transferStandard
(address,address,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._transferStandard
(address,address,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._transferFromExcluded(address,address,uint25
6).tTransferAmount (YummyToken.sol#782)
```



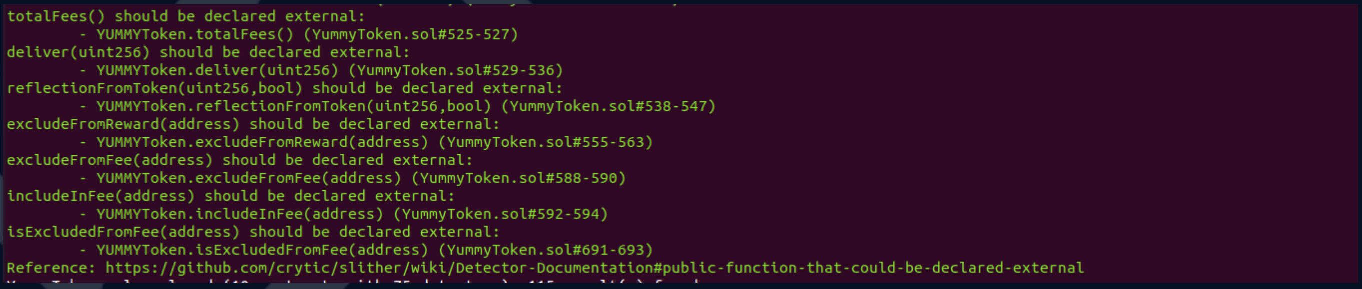
```
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._transferStandard(add
ress,address,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._transferStandard(a
ddress,address,uint256).tTransferAmount (YummyToken.sol#763)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._transferFromExcluded(address,
address,uint256).tTransferAmount (YummyToken.sol#782)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._getValues(uint256).tTransferAmount (YummyTo
ken.sol#619)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._getValues(uint256)
.tTransferAmount (YummyToken.sol#619)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._getValues(uint25
6).tTransferAmount (YummyToken.sol#619)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._transferToExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._transferToExcluded(address,address,uint256)
.tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._getTValues(uint256).
tTransferAmount (YummyToken.sol#627)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._transferToExcluded
(address,address,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._transferToExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._transferBothExcluded(address,
address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._getValues(uint256)
.tTransferAmount (YummyToken.sol#619)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._transferBothExcl
uded(address,address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._transferToExcluded
(address,address,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._getTValues(uint256).tTransfer
Amount (YummyToken.sol#627)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._transferBothExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._getValues(uint256).rTransferAmount (YummyToken.sol#620) is too similar to YUMMYToken._transferBothExcluded(address,address,uint25
6).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._getValues(uint256).t
TransferAmount (YummyToken.sol#619)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._transferBothExcl
uded(address,address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._transferFromExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#782)
```



```
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._transferToExcluded(a
ddress,address,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._getTValues(uint2
56).tTransferAmount (YummyToken.sol#627)
Variable YUMMYToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (YummyToken.sol#635) is too similar to YUMMYToken._getTValues(uint256
).tTransferAmount (YummyToken.sol#627)
Variable YUMMYToken._transferBothExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#578) is too similar to YUMMYToken._getTValues(uint2
56).tTransferAmount (YummyToken.sol#627)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._getValues(uint256).tTransferA
mount (YummyToken.sol#619)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._transferBothExclud
ed(address,address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken.reflectionFromToken(uint256,bool).rTransferAmount (YummyToken.sol#544) is too similar to YUMMYToken._transferToExcluded(address,ad
dress,uint256).tTransferAmount (YummyToken.sol#772)
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._transferBothExcluded
(address,address,uint256).tTransferAmount (YummyToken.sol#578)
Variable YUMMYToken._transferToExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#772) is too similar to YUMMYToken._getTValues(uint256
).tTransferAmount (YummyToken.sol#627)
Variable YUMMYToken._transferStandard(address,address,uint256).rTransferAmount (YummyToken.sol#763) is too similar to YUMMYToken._transferFromExcluded
(address,address,uint256).tTransferAmount (YummyToken.sol#782)
Variable YUMMYToken._transferFromExcluded(address,address,uint256).rTransferAmount (YummyToken.sol#782) is too similar to YUMMYToken._getValues(uint25
6).tTransferAmount (YummyToken.sol#619)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

YUMMYToken.slitherConstructorVariables() (YummyToken.sol#414-794) uses literals with too many digits:
        - _maxTxAmount = 1000000000 * 10 ** 1 * 10 ** 9 (YummyToken.sol#445)
YUMMYToken.slitherConstructorConstantVariables() (YummyToken.sol#414-794) uses literals with too many digits:
        - _tTotal = 100000000000 * 10 ** 1 * 10 ** 9 (YummyToken.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

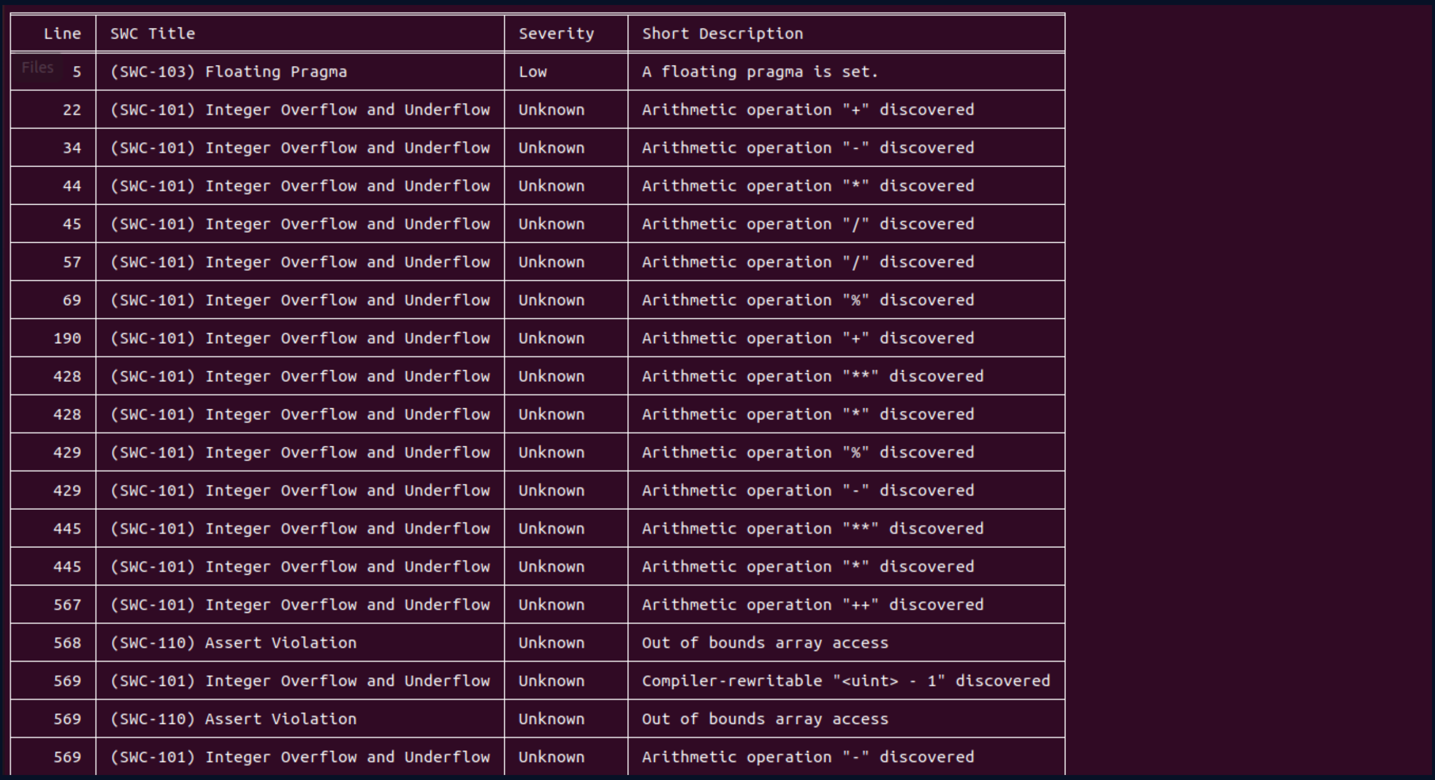
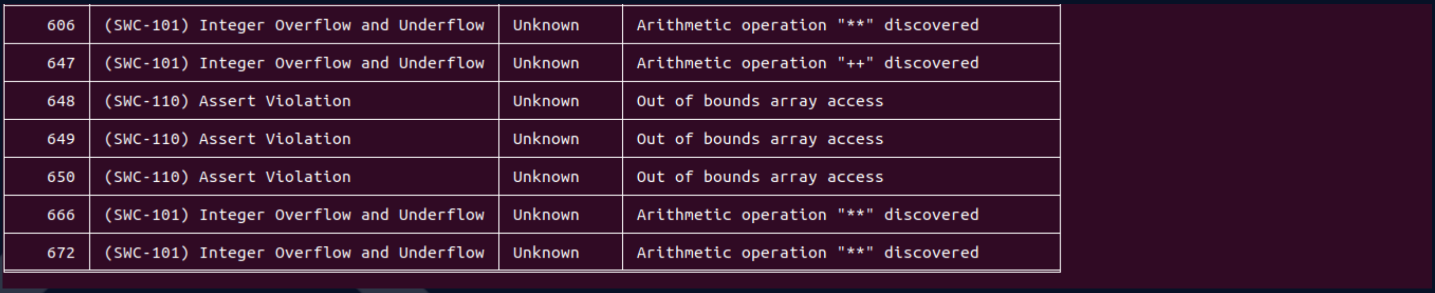


```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (YummyToken.sol#168-171)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (YummyToken.sol#173-177)
getUnlockTime() should be declared external:
        - Ownable.getUnlockTime() (YummyToken.sol#179-181)
getTime() should be declared external:
        - Ownable.getTime() (YummyToken.sol#183-185)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (YummyToken.sol#187-192)
unlock() should be declared external:
        - Ownable.unlock() (YummyToken.sol#194-199)
name() should be declared external:
        - YUMMYToken.name() (YummyToken.sol#465-467)
symbol() should be declared external:
        - YUMMYToken.symbol() (YummyToken.sol#469-471)
decimals() should be declared external:
        - YUMMYToken.decimals() (YummyToken.sol#473-475)
totalSupply() should be declared external:
        - YUMMYToken.totalSupply() (YummyToken.sol#477-479)
balanceOf(address) should be declared external:
        - YUMMYToken.balanceOf(address) (YummyToken.sol#481-484)
transfer(address,uint256) should be declared external:
        - YUMMYToken.transfer(address,uint256) (YummyToken.sol#486-489)
transferForeignToken(address,address) should be declared external:
        - YUMMYToken.transferForeignToken(address,address) (YummyToken.sol#491-494)
allowance(address,address) should be declared external:
        - YUMMYToken.allowance(address,address) (YummyToken.sol#496-498)
approve(address,uint256) should be declared external:
        - YUMMYToken.approve(address,uint256) (YummyToken.sol#500-503)
transferFrom(address,address,uint256) should be declared external:
        - YUMMYToken.transferFrom(address,address,uint256) (YummyToken.sol#505-509)
increaseAllowance(address,uint256) should be declared external:
        - YUMMYToken.increaseAllowance(address,uint256) (YummyToken.sol#511-514)
decreaseAllowance(address,uint256) should be declared external:
        - YUMMYToken.decreaseAllowance(address,uint256) (YummyToken.sol#516-519)
isExcludedFromReward(address) should be declared external:
        - YUMMYToken.isExcludedFromReward(address) (YummyToken.sol#521-523)
```



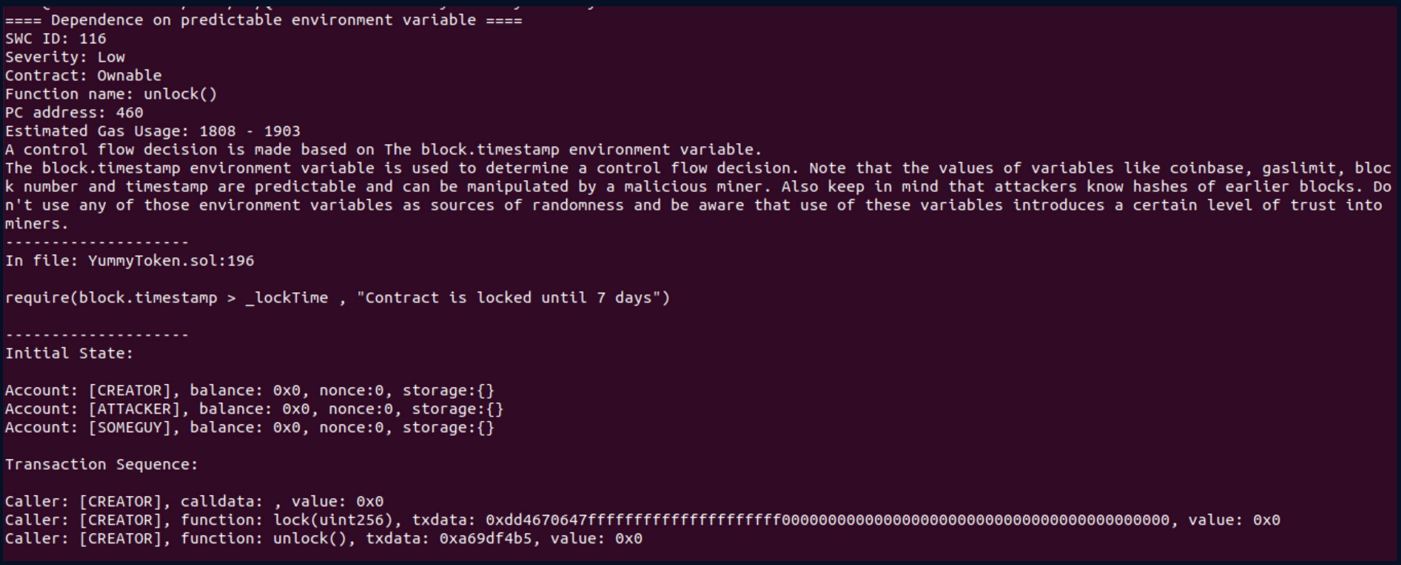
```
totalFees() should be declared external:
        - YUMMYToken.totalFees() (YummyToken.sol#525-527)
deliver(uint256) should be declared external:
        - YUMMYToken.deliver(uint256) (YummyToken.sol#529-536)
reflectionFromToken(uint256,bool) should be declared external:
        - YUMMYToken.reflectionFromToken(uint256,bool) (YummyToken.sol#538-547)
excludeFromReward(address) should be declared external:
        - YUMMYToken.excludeFromReward(address) (YummyToken.sol#555-563)
excludeFromFee(address) should be declared external:
        - YUMMYToken.excludeFromFee(address) (YummyToken.sol#588-590)
includeInFee(address) should be declared external:
        - YUMMYToken.includeInFee(address) (YummyToken.sol#592-594)
isExcludedFromFee(address) should be declared external:
        - YUMMYToken.isExcludedFromFee(address) (YummyToken.sol#691-693)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Mythx

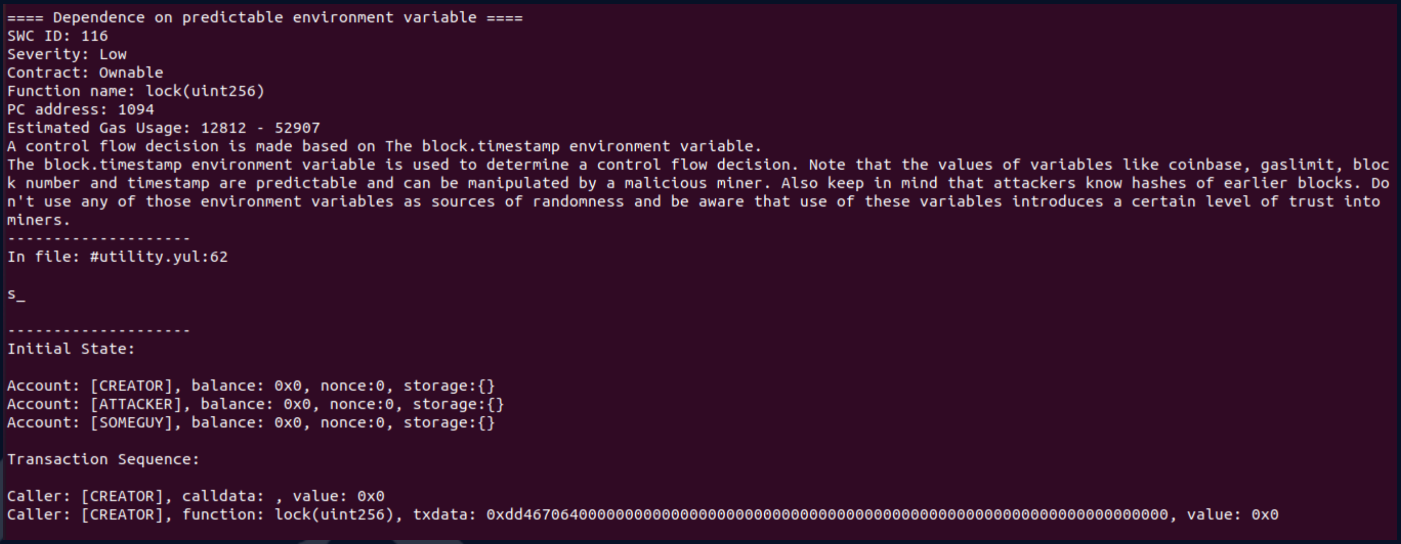



Line	SWC Title	Severity	Short Description
5	(SWC-103) Floating Pragma	Low	A floating pragma is set.
22	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
34	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
45	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
57	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
69	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
190	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
428	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
428	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
429	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
429	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
445	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
445	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
567	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
568	(SWC-110) Assert Violation	Unknown	Out of bounds array access
569	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
569	(SWC-110) Assert Violation	Unknown	Out of bounds array access
569	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
606	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
647	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
648	(SWC-110) Assert Violation	Unknown	Out of bounds array access
649	(SWC-110) Assert Violation	Unknown	Out of bounds array access
650	(SWC-110) Assert Violation	Unknown	Out of bounds array access
666	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
672	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered

Mythril

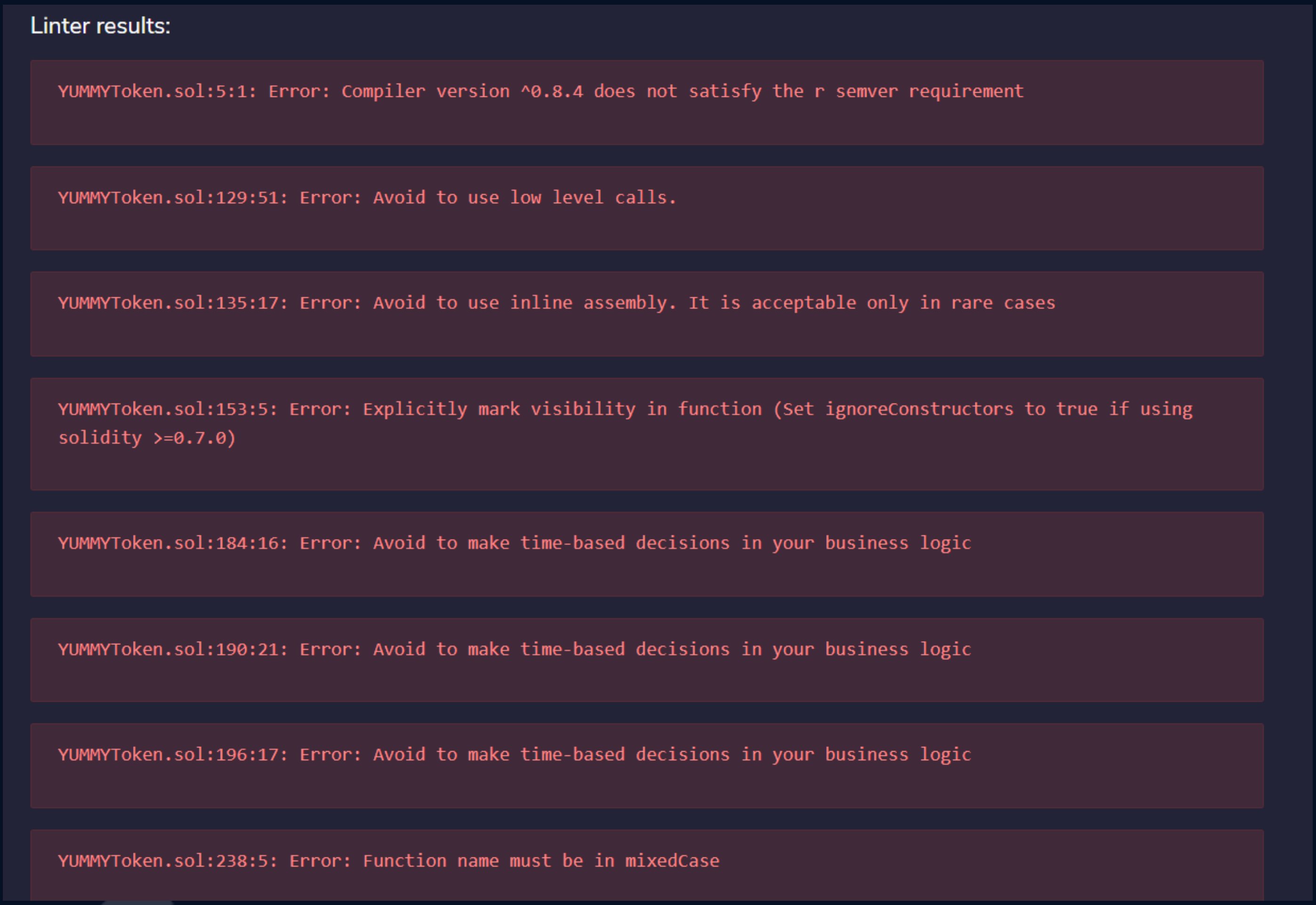


```
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Ownable
Function name: unlock()
PC address: 460
Estimated Gas Usage: 1808 - 1903
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, bloc
k number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Do
n'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.
--------------------
In file: YummyToken.sol:196

require(block.timestamp > _lockTime , "Contract is locked until 7 days")

--------------------
Initial State:

Account: [CREATOR], balance: 0x0, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: lock(uint256), txdata: 0xdd4670647fffffffffffffffffffff000000000000000000000000000000000000000000, value: 0x0
Caller: [CREATOR], function: unlock(), txdata: 0xa69df4b5, value: 0x0
```



```
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Ownable
Function name: lock(uint256)
PC address: 1094
Estimated Gas Usage: 12812 - 52907
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, bloc
k number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Do
n'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.
--------------------
In file: #utility.yul:62

s_

--------------------
Initial State:

Account: [CREATOR], balance: 0x0, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: lock(uint256), txdata: 0xdd4670640000000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

Solhint



Linter results:

```
YUMMYToken.sol:5:1: Error: Compiler version ^0.8.4 does not satisfy the r semver requirement
```

```
YUMMYToken.sol:129:51: Error: Avoid to use low level calls.
```

```
YUMMYToken.sol:135:17: Error: Avoid to use inline assembly. It is acceptable only in rare cases
```

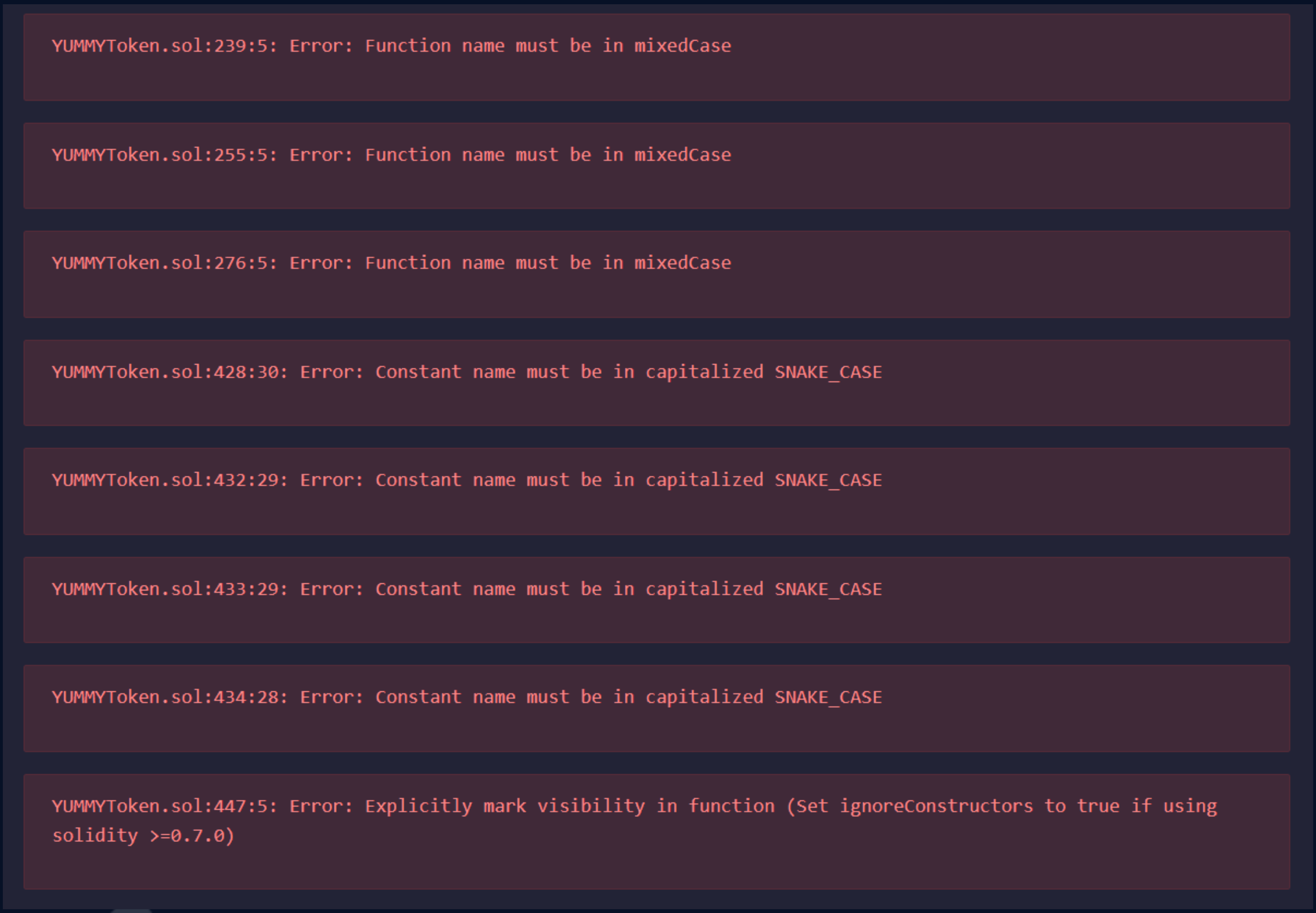
```
YUMMYToken.sol:153:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using
solidity >=0.7.0)
```

```
YUMMYToken.sol:184:16: Error: Avoid to make time-based decisions in your business logic
```

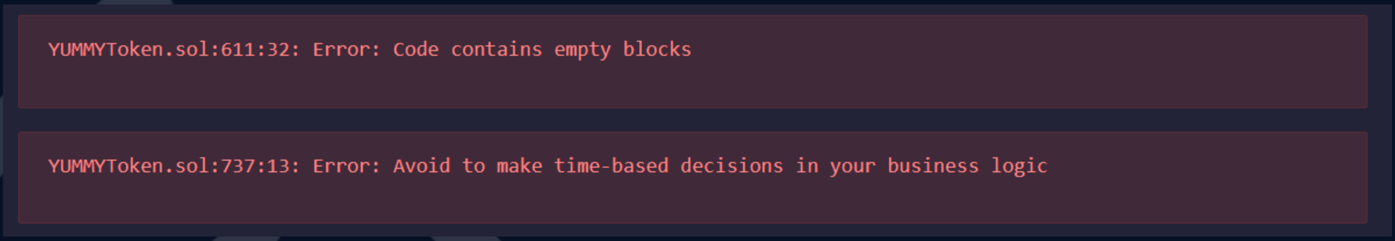
```
YUMMYToken.sol:190:21: Error: Avoid to make time-based decisions in your business logic
```

```
YUMMYToken.sol:196:17: Error: Avoid to make time-based decisions in your business logic
```

```
YUMMYToken.sol:238:5: Error: Function name must be in mixedCase
```



```
YUMMYToken.sol:239:5: Error: Function name must be in mixedCase

YUMMYToken.sol:255:5: Error: Function name must be in mixedCase

YUMMYToken.sol:276:5: Error: Function name must be in mixedCase

YUMMYToken.sol:428:30: Error: Constant name must be in capitalized SNAKE_CASE

YUMMYToken.sol:432:29: Error: Constant name must be in capitalized SNAKE_CASE

YUMMYToken.sol:433:29: Error: Constant name must be in capitalized SNAKE_CASE

YUMMYToken.sol:434:28: Error: Constant name must be in capitalized SNAKE_CASE

YUMMYToken.sol:447:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using
solidity >=0.7.0)
```



```
YUMMYToken.sol:611:32: Error: Code contains empty blocks

YUMMYToken.sol:737:13: Error: Avoid to make time-based decisions in your business logic
```

Closing Summary

Overall, in the Final audit, there are 2 low severity issues associated with the token. It is recommended to fix these before deployment. Overall the optimization issue is neglectable. No instances of Integer Overflow and Underflow, Reentrancy, or any other major vulnerabilities are found in the contract.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Yummy Token. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Yummy Token team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
September, 2021

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com