QuillAudits

Audit Report
May, 2022

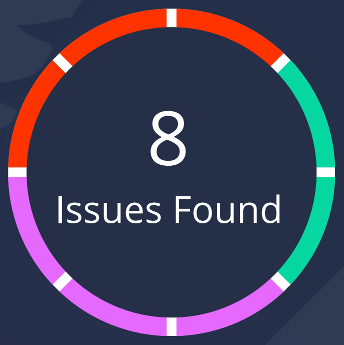
For

THE GAMBLE KINGDOM

Table of Content

Executive Summary

Project Name	The Gamble Kingdom
Overview	The Gamble Kingdom (TGK) team is building a unique virtual world where poker players can buy and hold their native token $TGK, collect TGK NFTs, participate in game and tournament play, and play-to-earn all within the metaverse kingdom. Players will be able to purchase, earn and collect the Kingdom's native token ($TGK) and use as currency for tournament, buy-ins, at-game stakes and wagers, as well as other tradeable features within the Kingdom.
Timeline	May 13th, 2022 to May 28th, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze and document The Gamble Kingdom smart contracts codebase for quality, security, and correctness.
Code Base	*https://github.com/shrishtieth/TGK/blob/main/contracts/token.sol* *https://github.com/shrishtieth/TGK/blob/main/contracts/taxDistributionContract.sol*
Commit Hash	2accb8e12a45b2134d21076fc1f544abbf35cc4a
Fixed In	6a02ea1ada0c580aa57a69a9f039f0fb3aed975a



🟥 High 🟨 Medium
🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	**3**	0	**2**	**3**

Types of Severities

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 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.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded

Severus.finance - Audit Report

- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

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

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

1. The blacklist account still can make transfer

Contract: Token.sol

Description

As per require check on line 125 the blackListed account still can make the transfer to other non blackListed account or vice versa.

```
require(isBlacklisted[sender]!= true || isBlacklisted[recipient]!= true,"Address Blacklisted");
```

Remediation

We recommend to put AND logic in require check instead of OR logic such that if any of the address from sender/recipent is blackListed then there should not be any transfer.

```
require(isBlacklisted[sender]!= true && isBlacklisted[recipient]!= true,"Address Blacklisted");
```

Status

Fixed

1. setTeamWalletAddress doesn't update the address correctly

Contract: taxDistributionContract.sol

Description

SetTeamWallet is setting the address of *investorWallet* instead of *teamWallet*.

Remediation

We recommend to put the correct wallet address to *teamWallet*.

Status

Fixed

3. setSlippage doesn’t update the addres

Contract: taxDistributionContract.sol

Description

SetSlippage function is setting the address of *slippage* instead of _*slippage*.

Remediation

We recommend to put the correct _*slippage* address to *slippage*.

Status

Fixed

Medium Severity Issues

No issues found

Low Severity Issues

4. setTeamWalletAddress doesn’t update the address correctly

Contract: taxDistributionContract.sol

Description

InitialInvestorPercentage and investorPercentage in there respective setter should be less than 10,000.

Remediation

We should to put a require check while setting the InitialInvestorPercentage and investorPercentage at the time of setting it.

Status

Fixed

5. Owner should be multisig

Description

We recommend to use multisig account address (gnosis-safe) for owner such that the pool creating is not been malicious in future and the decentralization is achieved in the system.

Status

Fixed

Informational Issues

6. Redundant code

Description

The code in if and else block of distributeTax is redundant. We recommend to remove the redundancy from the code.

Status

Fixed

7.1. Gas optimizations

Contract: taxDistributionContract.sol

Description

Whenever swapTokensForEth is called the approval to uniswapV2Router With INTMAX is made which is a wastage of gas.

Remediation

We recommend to make the approval individually in other function and that should be called at the time of setting the uniswapV2Router.

```
function setRouterAddress(address router) external onlyOwner{
      uniswapV2Router = IUniswapV2Router02(router);
   IERC20(TGKToken).approve(address(uniswapV2Router), 2**256 - 1);
      emit RouterUpdated(router);
  }
```

Status

Fixed

7.2. Gas optimizations

Contract: taxDistributionContract.sol

Description

At the time of calculating investorAmount and priceImpact the denominator 10,000 is constant so we recommend to use unchecked flag at the time of dividing as no need to use inbuilt safeMath wrappers and waste the gas.

Status

Fixed

8. Missing netspec comments

Recommendation

We recommend adding netspec comments for each method and variables for better readability and understanding of code.

Status

Fixed

Functional Tests (RINKEBY)

Contracts

- ✓ Token - 0xefcDc0fc735a3cBb3a1b9F7D75da65507Af06498
- ✓ TaxDistribution - 0x9f385B638efEAd7BfEB3a32472CA94bAdadE93c8

Transactions

- ✓ **Burn**
  https://rinkeby.etherscan.io/tx/0x488a5ab64925fad4d22d1e48e2791938813ea352d77b40178c38c608a27df9ad

- ✓ **Airdrop to 2 addresses**
  https://rinkeby.etherscan.io/tx/0x18000213c91ac72c51caa10ae80ac52fcf37719a0ecb8f5ab6e4b0537c4d9552

- ✓ **updateMaxBuy**
  https://rinkeby.etherscan.io/tx/0x789b3d19a000cceab7fbb9de7fae1f8d2b94bf9e118bcf24d6a54c168c13f379

- ✓ **ONLYOWNER can update the functionality**
  https://rinkeby.etherscan.io/tx/0x5e0ea9ae0603f5256452602b26b3bb10598d4a2478c8a5f6b425c0da172e733c

- ✓ **updateMaxSell**
  https://rinkeby.etherscan.io/tx/0xf7324598d7e80544da722e69d9e159f463e50a07a6153cf529195535ad40a882

- ✓ **updateBlackList**
  https://rinkeby.etherscan.io/tx/0xc18caf67fb2cee5f7ac764d1358464b9fe5c1bc6019b177e777ddad6a81add6a

- ✓ **UpdateEpoch**
  https://rinkeby.etherscan.io/tx/0x8447ca5b2b58fa663da172b23238cb2b27de654b967f562c326c50936e3020d3

✓ **Approve router contract**
https://rinkeby.etherscan.io/tx/0x22053d3a19d3216f4a8dd110236379ff3466ba0952f0f34cadf4d1cdeb3c54dd

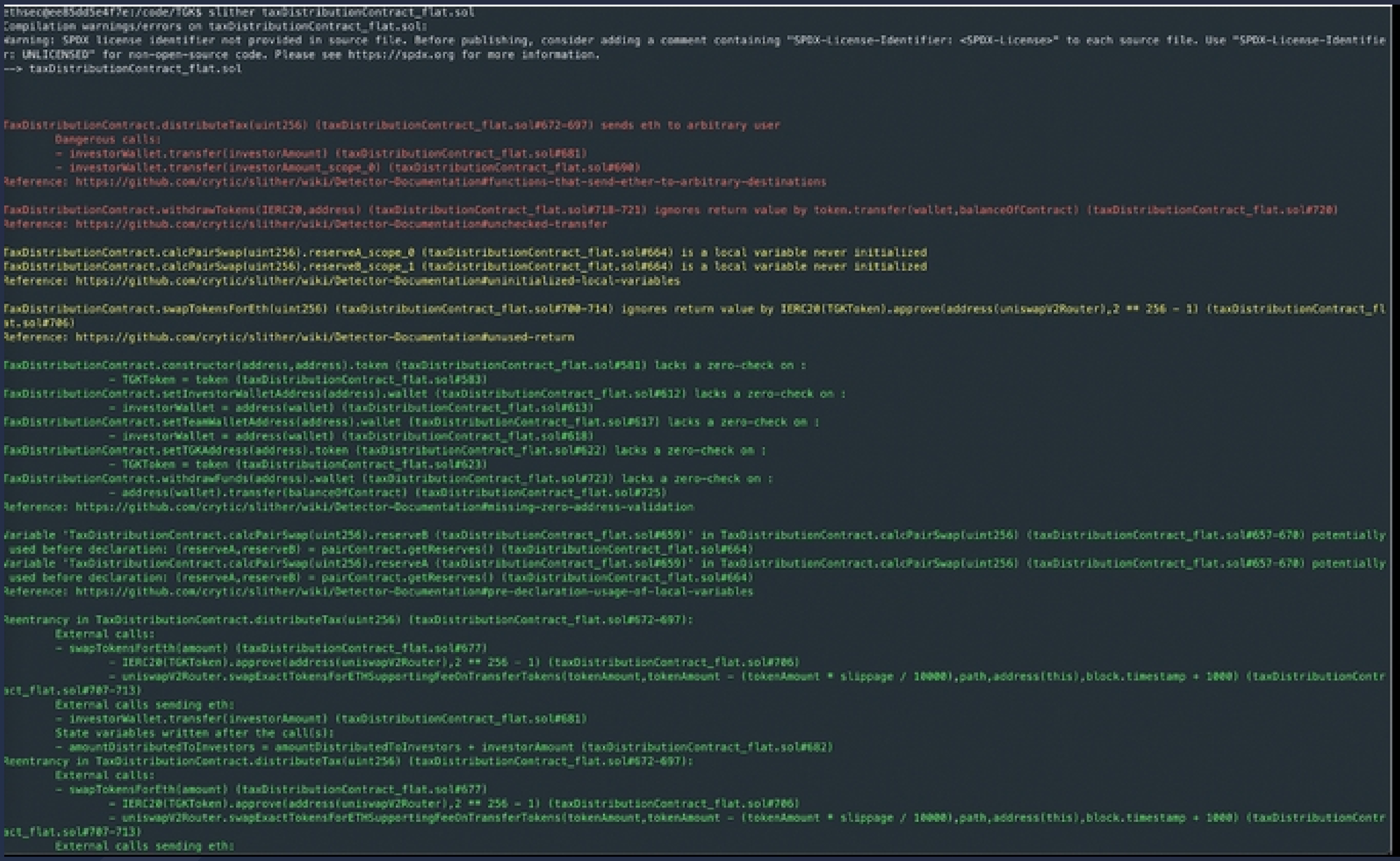
✓ **Amount exceed max sell value**
https://rinkeby.etherscan.io/tx/0x1a1b76eacab26bda8be1279a5a67aa1d89fa4fddd4a660bd7a2bff86df480227

✓ **Transfer funds to tax contract**
https://rinkeby.etherscan.io/tx/0xba2f72b89ecb2b286356390a95d3f9e3b1d9c0269176800a27a19913fd7b5772

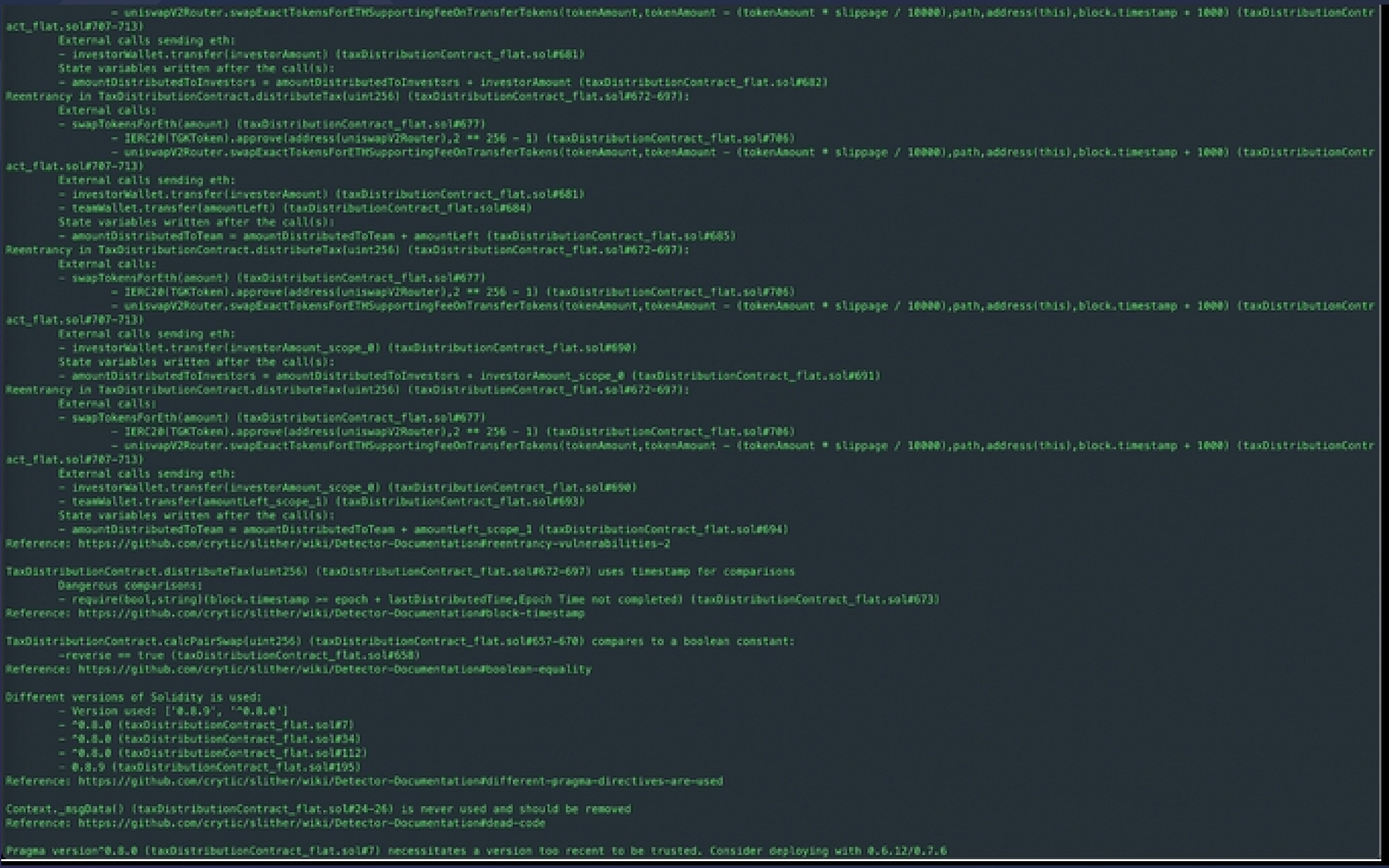
✓ **setInvestorThresholdAmount**
https://rinkeby.etherscan.io/tx/0x5b163db2048fa860f52da0beb1158faaa058548c1f1ca50594793b10ee40cbbf

✓ **setTeamWalletAddress (Not updated Failed)**
https://rinkeby.etherscan.io/tx/0x528c48cd93add3c8ab90977e0953406ac745f122212de9097a3e8a61526ccfee

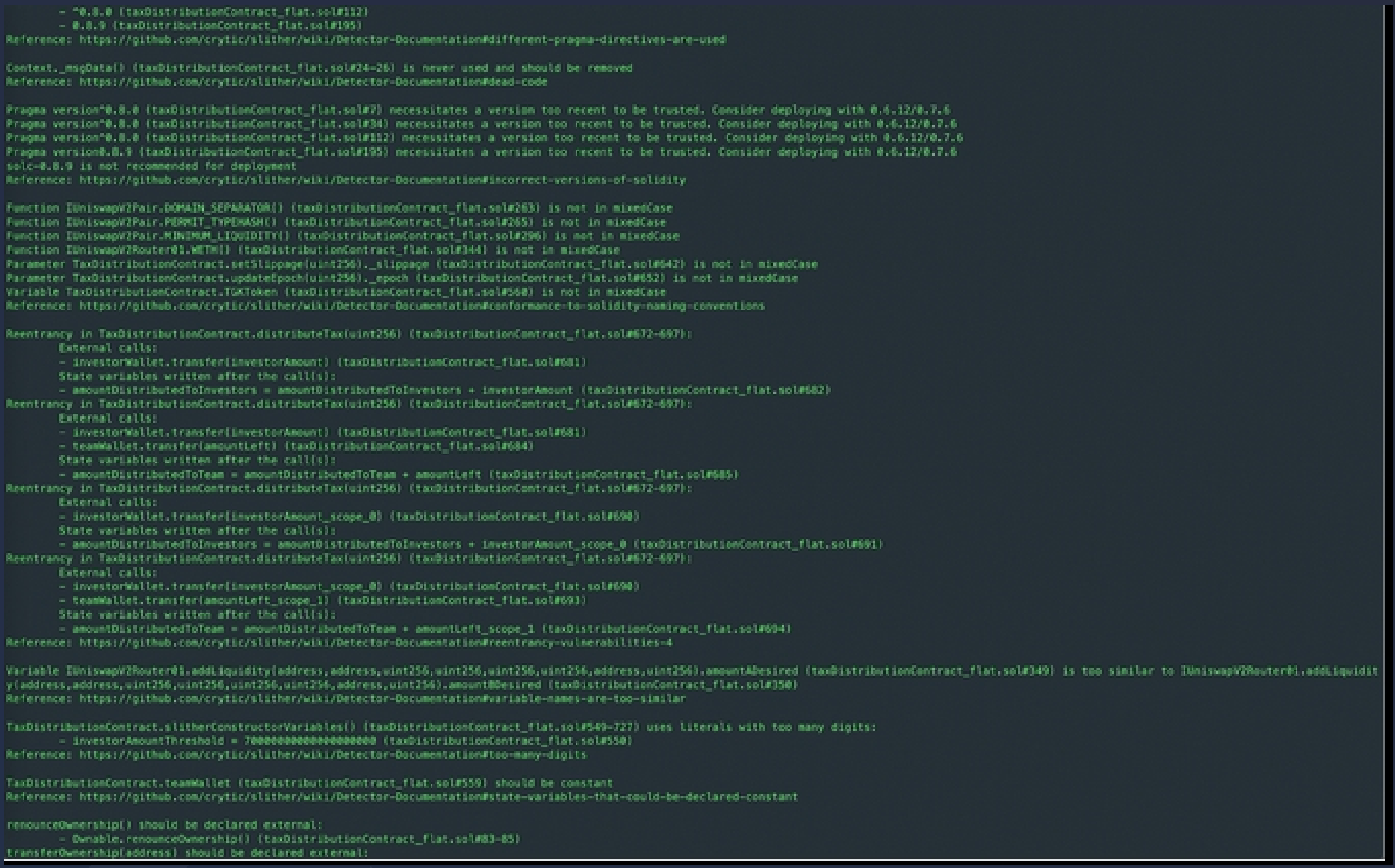
Automated Tests



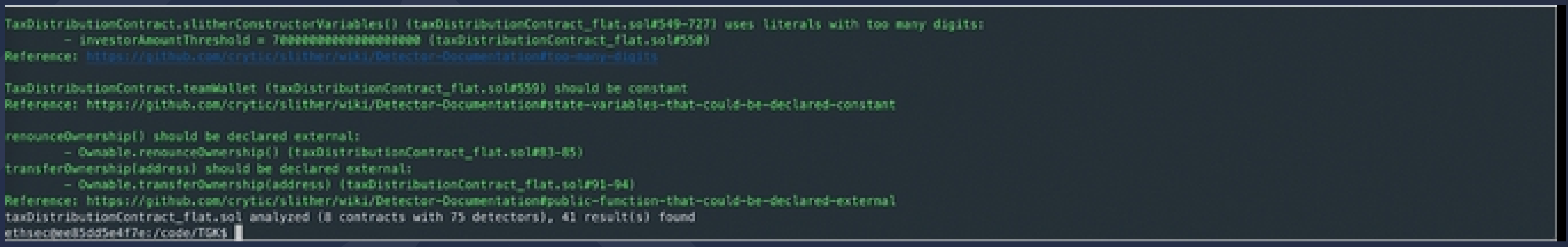
```
ethsec@ee85dd5e4f7e:/code/TGK$ slither taxDistributionContract_flat.sol
Compilation warnings/errors on taxDistributionContract_flat.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifie
r: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> taxDistributionContract_flat.sol



TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697) sends eth to arbitrary user
        Dangerous calls:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        - investorWallet.transfer(investorAmount_scope_0) (taxDistributionContract_flat.sol#690)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

TaxDistributionContract.withdrawTokens(IERC20,address) (taxDistributionContract_flat.sol#718-721) ignores return value by token.transfer(wallet,balanceOfContract) (taxDistributionContract_flat.sol#720)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

TaxDistributionContract.calcPairSwap(uint256).reserveA_scope_0 (taxDistributionContract_flat.sol#664) is a local variable never initialized
TaxDistributionContract.calcPairSwap(uint256).reserveB_scope_1 (taxDistributionContract_flat.sol#664) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

TaxDistributionContract.swapTokensForEth(uint256) (taxDistributionContract_flat.sol#700-714) ignores return value by IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_fl
at.sol#706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TaxDistributionContract.constructor(address,address).token (taxDistributionContract_flat.sol#581) lacks a zero-check on :
                - TGKToken = token (taxDistributionContract_flat.sol#583)
TaxDistributionContract.setInvestorWalletAddress(address).wallet (taxDistributionContract_flat.sol#612) lacks a zero-check on :
                - investorWallet = address(wallet) (taxDistributionContract_flat.sol#613)
TaxDistributionContract.setTeamWalletAddress(address).wallet (taxDistributionContract_flat.sol#617) lacks a zero-check on :
                - investorWallet = address(wallet) (taxDistributionContract_flat.sol#618)
TaxDistributionContract.setTGKAddress(address).token (taxDistributionContract_flat.sol#622) lacks a zero-check on :
                - TGKToken = token (taxDistributionContract_flat.sol#623)
TaxDistributionContract.withdrawFunds(address).wallet (taxDistributionContract_flat.sol#723) lacks a zero-check on :
                - address(wallet).transfer(balanceOfContract) (taxDistributionContract_flat.sol#725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'TaxDistributionContract.calcPairSwap(uint256).reserveB (taxDistributionContract_flat.sol#659)' in TaxDistributionContract.calcPairSwap(uint256) (taxDistributionContract_flat.sol#657-670) potentially
 used before declaration: (reserveA,reserveB) = pairContract.getReserves() (taxDistributionContract_flat.sol#664)
Variable 'TaxDistributionContract.calcPairSwap(uint256).reserveA (taxDistributionContract_flat.sol#659)' in TaxDistributionContract.calcPairSwap(uint256) (taxDistributionContract_flat.sol#657-670) potentially
 used before declaration: (reserveA,reserveB) = pairContract.getReserves() (taxDistributionContract_flat.sol#664)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - swapTokensForEth(amount) (taxDistributionContract_flat.sol#677)
                - IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_flat.sol#706)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        State variables written after the call(s):
        - amountDistributedToInvestors = amountDistributedToInvestors + investorAmount (taxDistributionContract_flat.sol#682)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - swapTokensForEth(amount) (taxDistributionContract_flat.sol#677)
                - IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_flat.sol#706)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
```



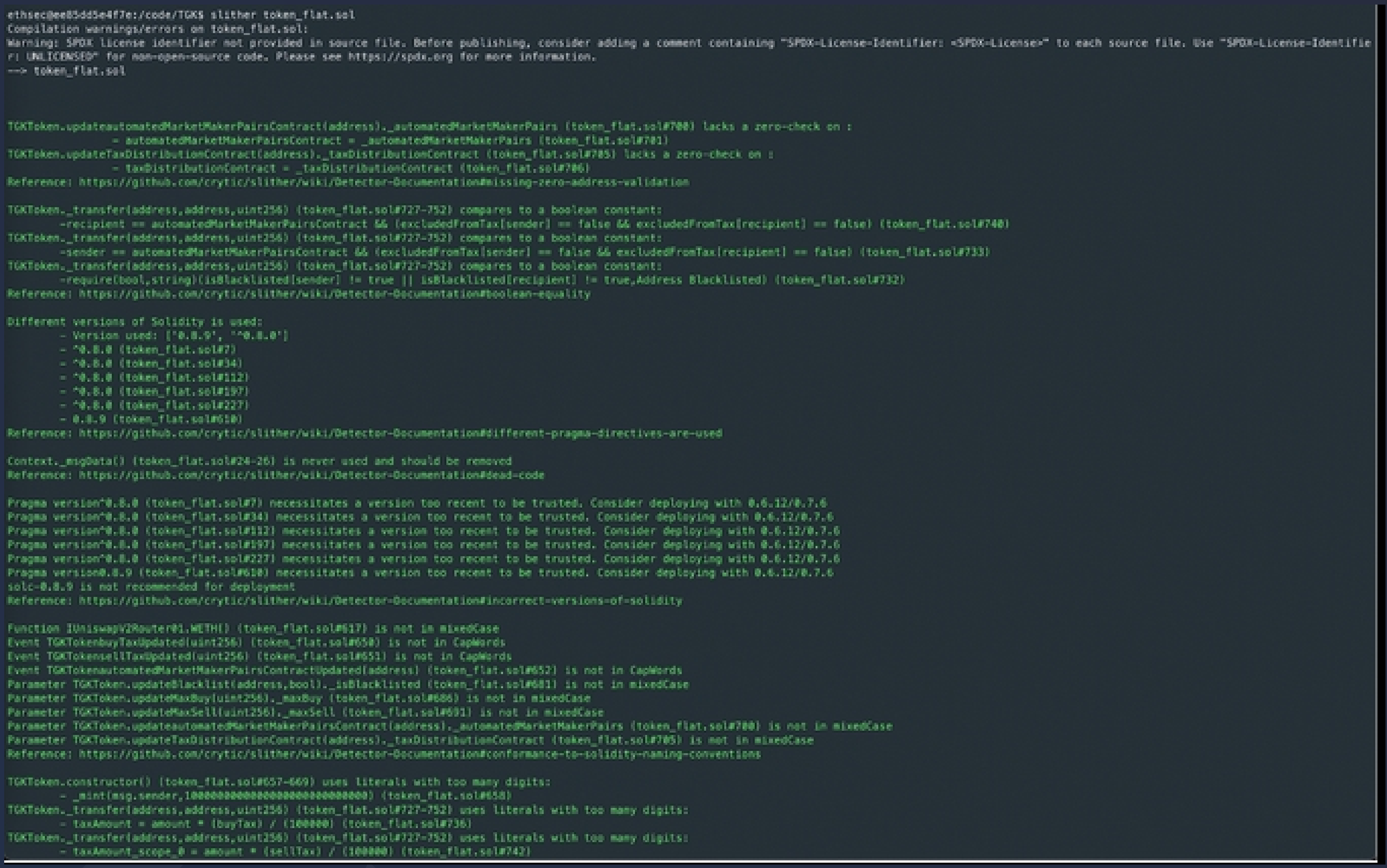
```
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        State variables written after the call(s):
        - amountDistributedToInvestors = amountDistributedToInvestors + investorAmount (taxDistributionContract_flat.sol#682)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - swapTokensForEth(amount) (taxDistributionContract_flat.sol#677)
                - IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_flat.sol#706)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        - teamWallet.transfer(amountLeft) (taxDistributionContract_flat.sol#684)
        State variables written after the call(s):
        - amountDistributedToTeam = amountDistributedToTeam + amountLeft (taxDistributionContract_flat.sol#685)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - swapTokensForEth(amount) (taxDistributionContract_flat.sol#677)
                - IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_flat.sol#706)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
        - investorWallet.transfer(investorAmount_scope_0) (taxDistributionContract_flat.sol#690)
        State variables written after the call(s):
        - amountDistributedToInvestors = amountDistributedToInvestors + investorAmount_scope_0 (taxDistributionContract_flat.sol#691)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - swapTokensForEth(amount) (taxDistributionContract_flat.sol#677)
                - IERC20(TGKToken).approve(address(uniswapV2Router),2 ** 256 - 1) (taxDistributionContract_flat.sol#706)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,tokenAmount - (tokenAmount * slippage / 10000),path,address(this),block.timestamp + 1000) (taxDistributionContr
act_flat.sol#707-713)
        External calls sending eth:
        - investorWallet.transfer(investorAmount_scope_0) (taxDistributionContract_flat.sol#690)
        - teamWallet.transfer(amountLeft_scope_1) (taxDistributionContract_flat.sol#693)
        State variables written after the call(s):
        - amountDistributedToTeam = amountDistributedToTeam + amountLeft_scope_1 (taxDistributionContract_flat.sol#694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= epoch + lastDistributedTime,Epoch Time not completed) (taxDistributionContract_flat.sol#673)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

TaxDistributionContract.calcPairSwap(uint256) (taxDistributionContract_flat.sol#657-670) compares to a boolean constant:
        -reverse == true (taxDistributionContract_flat.sol#658)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (taxDistributionContract_flat.sol#7)
        - ^0.8.0 (taxDistributionContract_flat.sol#34)
        - ^0.8.0 (taxDistributionContract_flat.sol#112)
        - 0.8.9 (taxDistributionContract_flat.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (taxDistributionContract_flat.sol#24-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (taxDistributionContract_flat.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
```



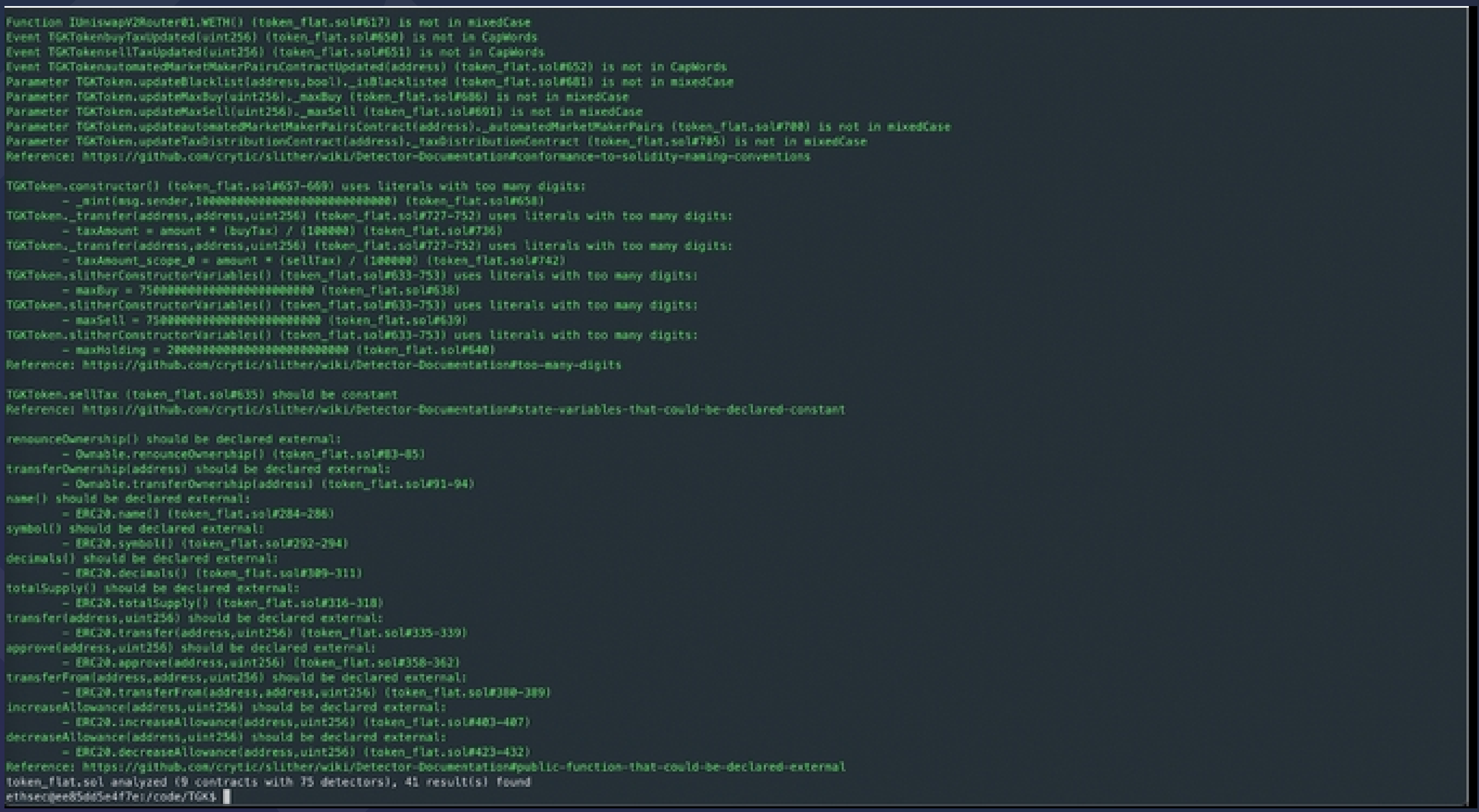
```
        - ^0.8.0 (taxDistributionContract_flat.sol#112)
        - 0.8.9 (taxDistributionContract_flat.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (taxDistributionContract_flat.sol#24-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (taxDistributionContract_flat.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (taxDistributionContract_flat.sol#34) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (taxDistributionContract_flat.sol#112) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (taxDistributionContract_flat.sol#195) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (taxDistributionContract_flat.sol#263) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (taxDistributionContract_flat.sol#265) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (taxDistributionContract_flat.sol#296) is not in mixedCase
Function IUniswapV2Router01.WETH() (taxDistributionContract_flat.sol#344) is not in mixedCase
Parameter TaxDistributionContract.setSlippage(uint256)._slippage (taxDistributionContract_flat.sol#642) is not in mixedCase
Parameter TaxDistributionContract.updateEpoch(uint256)._epoch (taxDistributionContract_flat.sol#652) is not in mixedCase
Variable TaxDistributionContract.TGKToken (taxDistributionContract_flat.sol#560) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        State variables written after the call(s):
        - amountDistributedToInvestors = amountDistributedToInvestors + investorAmount (taxDistributionContract_flat.sol#682)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - investorWallet.transfer(investorAmount) (taxDistributionContract_flat.sol#681)
        - teamWallet.transfer(amountLeft) (taxDistributionContract_flat.sol#684)
        State variables written after the call(s):
        - amountDistributedToTeam = amountDistributedToTeam + amountLeft (taxDistributionContract_flat.sol#685)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - investorWallet.transfer(investorAmount_scope_0) (taxDistributionContract_flat.sol#690)
        State variables written after the call(s):
        - amountDistributedToInvestors = amountDistributedToInvestors + investorAmount_scope_0 (taxDistributionContract_flat.sol#691)
Reentrancy in TaxDistributionContract.distributeTax(uint256) (taxDistributionContract_flat.sol#672-697):
        External calls:
        - investorWallet.transfer(investorAmount_scope_0) (taxDistributionContract_flat.sol#690)
        - teamWallet.transfer(amountLeft_scope_1) (taxDistributionContract_flat.sol#693)
        State variables written after the call(s):
        - amountDistributedToTeam = amountDistributedToTeam + amountLeft_scope_1 (taxDistributionContract_flat.sol#694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (taxDistributionContract_flat.sol#349) is too similar to IUniswapV2Router01.addLiquidit
y(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (taxDistributionContract_flat.sol#350)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

TaxDistributionContract.slitherConstructorVariables() (taxDistributionContract_flat.sol#549-727) uses literals with too many digits:
        - investorAmountThreshold = 7000000000000000000000 (taxDistributionContract_flat.sol#550)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TaxDistributionContract.teamWallet (taxDistributionContract_flat.sol#559) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (taxDistributionContract_flat.sol#83-85)
transferOwnership(address) should be declared external:
```



```
TaxDistributionContract.slitherConstructorVariables() (taxDistributionContract_flat.sol#549-727) uses literals with too many digits:
        - investorAmountThreshold = 7000000000000000000000 (taxDistributionContract_flat.sol#550)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TaxDistributionContract.teamWallet (taxDistributionContract_flat.sol#559) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (taxDistributionContract_flat.sol#83-85)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (taxDistributionContract_flat.sol#91-94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
taxDistributionContract_flat.sol analyzed (8 contracts with 75 detectors), 41 result(s) found
ethsec@ee85dd5e4f7e:/code/TGK$ 
```



```
ethsec@ee85dd5e4f7e:/code/TGK$ slither token_flat.sol
Compilation warnings/errors on token_flat.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifie
r: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> token_flat.sol



TGKToken.updateautomatedMarketMakerPairsContract(address)._automatedMarketMakerPairs (token_flat.sol#700) lacks a zero-check on :
                - automatedMarketMakerPairsContract = _automatedMarketMakerPairs (token_flat.sol#701)
TGKToken.updateTaxDistributionContract(address)._taxDistributionContract (token_flat.sol#705) lacks a zero-check on :
                - taxDistributionContract = _taxDistributionContract (token_flat.sol#706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) compares to a boolean constant:
        -recipient == automatedMarketMakerPairsContract && (excludedFromTax[sender] == false && excludedFromTax[recipient] == false) (token_flat.sol#740)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) compares to a boolean constant:
        -sender == automatedMarketMakerPairsContract && (excludedFromTax[sender] == false && excludedFromTax[recipient] == false) (token_flat.sol#733)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) compares to a boolean constant:
        -require(bool,string)(isBlacklisted[sender] != true || isBlacklisted[recipient] != true,Address Blacklisted) (token_flat.sol#732)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (token_flat.sol#7)
        - ^0.8.0 (token_flat.sol#34)
        - ^0.8.0 (token_flat.sol#112)
        - ^0.8.0 (token_flat.sol#197)
        - ^0.8.0 (token_flat.sol#227)
        - 0.8.9 (token_flat.sol#610)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (token_flat.sol#24-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (token_flat.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token_flat.sol#34) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token_flat.sol#112) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token_flat.sol#197) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token_flat.sol#227) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (token_flat.sol#610) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (token_flat.sol#617) is not in mixedCase
Event TGKTokenbuyTaxUpdated(uint256) (token_flat.sol#650) is not in CapWords
Event TGKTokensellTaxUpdated(uint256) (token_flat.sol#651) is not in CapWords
Event TGKTokenautomatedMarketMakerPairsContractUpdated(address) (token_flat.sol#652) is not in CapWords
Parameter TGKToken.updateBlacklist(address,bool)._isBlacklisted (token_flat.sol#681) is not in mixedCase
Parameter TGKToken.updateMaxBuy(uint256)._maxBuy (token_flat.sol#686) is not in mixedCase
Parameter TGKToken.updateMaxSell(uint256)._maxSell (token_flat.sol#691) is not in mixedCase
Parameter TGKToken.updateautomatedMarketMakerPairsContract(address)._automatedMarketMakerPairs (token_flat.sol#700) is not in mixedCase
Parameter TGKToken.updateTaxDistributionContract(address)._taxDistributionContract (token_flat.sol#705) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

TGKToken.constructor() (token_flat.sol#657-669) uses literals with too many digits:
        - _mint(msg.sender,1000000000000000000000000000) (token_flat.sol#658)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) uses literals with too many digits:
        - taxAmount = amount * (buyTax) / (100000) (token_flat.sol#736)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) uses literals with too many digits:
        - taxAmount_scope_0 = amount * (sellTax) / (100000) (token_flat.sol#742)
```



```
Function IUniswapV2Router01.WETH() (token_flat.sol#617) is not in mixedCase
Event TGKTokenbuyTaxUpdated(uint256) (token_flat.sol#650) is not in CapWords
Event TGKTokensellTaxUpdated(uint256) (token_flat.sol#651) is not in CapWords
Event TGKTokenautomatedMarketMakerPairsContractUpdated(address) (token_flat.sol#652) is not in CapWords
Parameter TGKToken.updateBlacklist(address,bool)._isBlacklisted (token_flat.sol#681) is not in mixedCase
Parameter TGKToken.updateMaxBuy(uint256)._maxBuy (token_flat.sol#686) is not in mixedCase
Parameter TGKToken.updateMaxSell(uint256)._maxSell (token_flat.sol#691) is not in mixedCase
Parameter TGKToken.updateautomatedMarketMakerPairsContract(address)._automatedMarketMakerPairs (token_flat.sol#700) is not in mixedCase
Parameter TGKToken.updateTaxDistributionContract(address)._taxDistributionContract (token_flat.sol#705) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

TGKToken.constructor() (token_flat.sol#657-669) uses literals with too many digits:
        - _mint(msg.sender,1000000000000000000000000000) (token_flat.sol#658)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) uses literals with too many digits:
        - taxAmount = amount * (buyTax) / (100000) (token_flat.sol#736)
TGKToken._transfer(address,address,uint256) (token_flat.sol#727-752) uses literals with too many digits:
        - taxAmount_scope_0 = amount * (sellTax) / (100000) (token_flat.sol#742)
TGKToken.slitherConstructorVariables() (token_flat.sol#633-753) uses literals with too many digits:
        - maxBuy = 7500000000000000000000000 (token_flat.sol#638)
TGKToken.slitherConstructorVariables() (token_flat.sol#633-753) uses literals with too many digits:
        - maxSell = 7500000000000000000000000 (token_flat.sol#639)
TGKToken.slitherConstructorVariables() (token_flat.sol#633-753) uses literals with too many digits:
        - maxHolding = 20000000000000000000000000 (token_flat.sol#640)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TGKToken.sellTax (token_flat.sol#635) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (token_flat.sol#83-85)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (token_flat.sol#91-94)
name() should be declared external:
        - ERC20.name() (token_flat.sol#284-286)
symbol() should be declared external:
        - ERC20.symbol() (token_flat.sol#292-294)
decimals() should be declared external:
        - ERC20.decimals() (token_flat.sol#309-311)
totalSupply() should be declared external:
        - ERC20.totalSupply() (token_flat.sol#316-318)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (token_flat.sol#335-339)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (token_flat.sol#358-362)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (token_flat.sol#380-389)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (token_flat.sol#403-407)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (token_flat.sol#423-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
token_flat.sol analyzed (9 contracts with 75 detectors), 41 result(s) found
ethsec@ee85dd5e4f7e:/code/TGK$
```

Results

A few major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are well written and adhere to guidelines.

Numerous issues were discovered in the initial audit. In the End, The Gamble Kingdom Team resolved all Issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Gamble Kingdom platform. 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 Gamble Kingdom team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2022

For

THE GAMBLE KINGDOM

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com