Smart Contract Security Audit V1

Quack Gold

https://quackgold.com/

21/11/2021

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

_

Table of Contents

Background

The purpose of the audit was to achieve the following:

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

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Website**: https://quackgold.com/
- **Twitter**: https://twitter.com/quack_gold
- **Telegram group:** https://t.me/quackgold
- **WhitePaper:** https://docs.quackgold.com/
- **Platform**: Binance Smart Chain
- **Contract Address**: 0x1efcbb4e3b73e7ca0c11b696bdc87ad7eadc79e1

Quack Gold Token ($QGOLD) With a total supply of 1 quadrillion, quack gold is designed with an inbuilt mechanism that burns 10% of every transaction into a pump wallet

Token Information

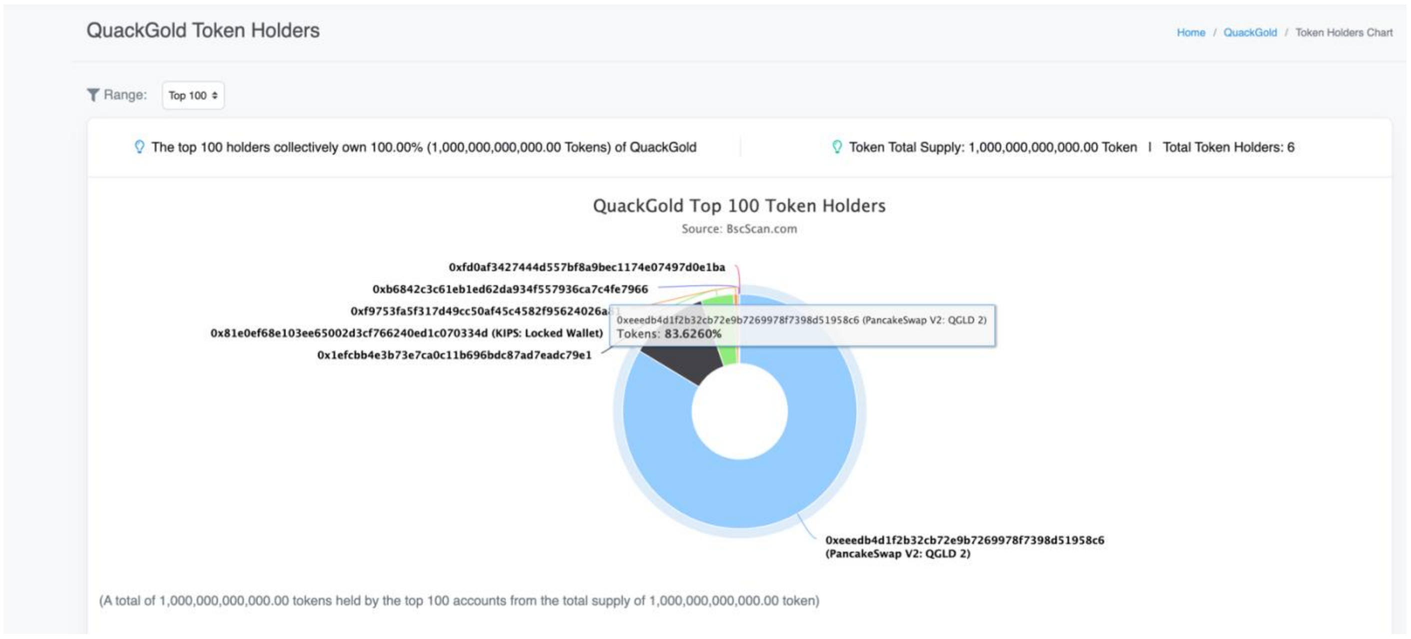
- Name: QGOLD
- Total Supply: 1,000,000,000,000
- Holders: 6 address
- Total transactions: 12

Contracts address deployed to test net (BSC)

Quack Gold token (QGOLD)contract on testnet.bsc (BSC Test Net)

https://testnet.bscscan.com/address/0x8c2db4e64c3d793fffd0ae9a4a70d829f1043906

QGOLD Token Distribution



QuackGold Token Holders

Home / QuackGold / Token Holders Chart

Range: Top 100

The top 100 holders collectively own 100.00% (1,000,000,000,000.00 Tokens) of QuackGold

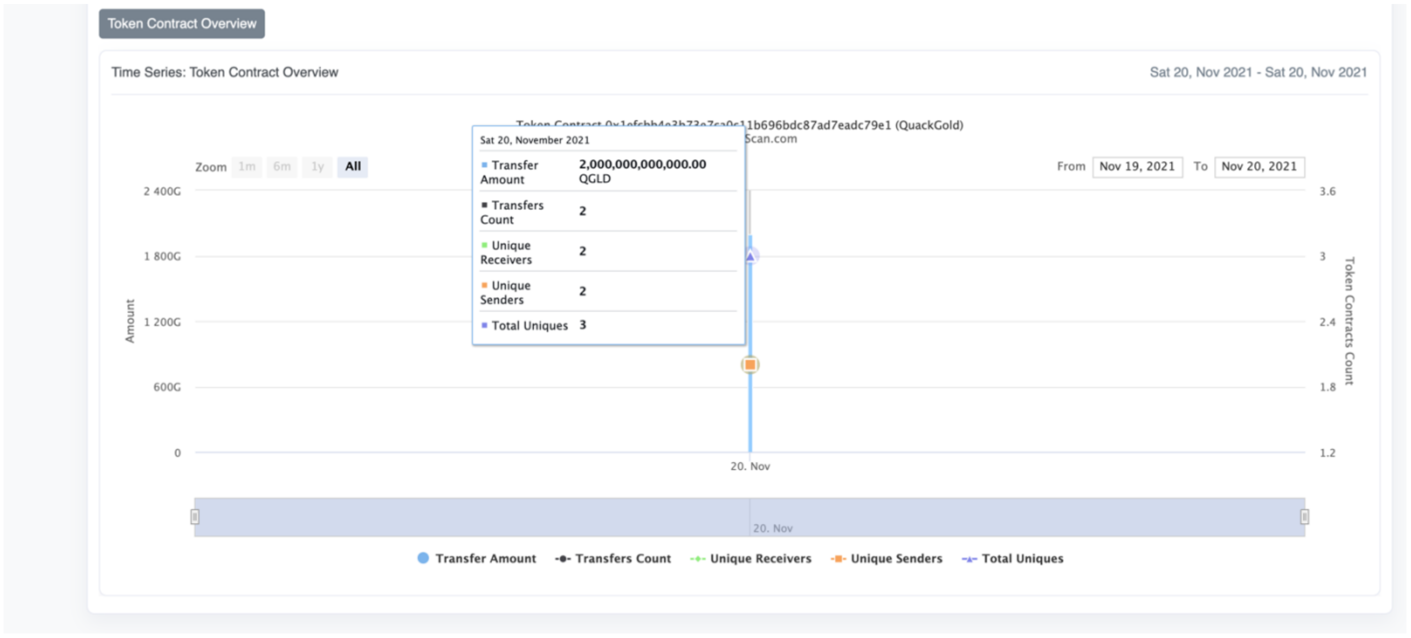
Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 6

QuackGold Top 100 Token Holders

Source: BscScan.com

(A total of 1,000,000,000,000.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000.00 token)

Contract Interaction Details



Token Contract Overview

Time Series: Token Contract Overview

Sat 20, Nov 2021 - Sat 20, Nov 2021

Token Contract 0x1efcbb4e3b73e7ca0c11b696bdc87ad7eadc79e1 (QuackGold)

Source: BscScan.com

Sat 20, November 2021

Transfer Amount	2,000,000,000,000.00 QGLD
Transfers Count	2
Unique Receivers	2
Unique Senders	2
Total Uniques	3

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 3 low, 2 very low-level issues and 1 note in all solidity files of the contract

The files:

QGOLD.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
QGOLD.sol	**45f85e1736b510305873efb79889edda47ccfb653878d66c2365bc15e2c3750e**	0x1efcbb4e3b73e7ca0c11b696bdc87ad7eadc79e1

- Contract: QGOLD
- Inherit: ERC20, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
_isBlacklisted	✓	Read / private	**Passed**
_marketingWalletAddress	✓	Read / private	**Passed**
automatedMarketMakerPairs	✓	Read / public	**Passed**

deadWallet	✓	Read / public	**Passed**
dividendTokenBalanceOf	✓	Read / public	**Passed**
dividendTracker	✓	Read / public	**Passed**
gasForProcessing	✓	Read / public	**Passed**
getAccountDividendsInfo	✓	Read / public	**Passed**
getAccountDividendsInfoAtIndex	✓	Read / public	**Passed**
getClaimWait	✓	Read / public	**Passed**
getLastProcessedIndex	✓	Read / public	**Passed**
getNumberOfDividendTokenHolders	✓	Read / public	**Passed**
getTotalDividendsDistributed	✓	Read / public	**Passed**
isExcludedFromFees	✓	Read / public	**Passed**
liquidityFee	✓	Read / public	**Passed**
marketingFee	✓	Read / public	**Passed**
swapTokensAtAmount	✓	Read / public	**Passed**
totalFees	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
USDTRewardsFee	✓	Read / public	**Passed**
USDT	✓	Read / public	**Passed**
withdrawableDividendOf	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
TransferFrom	✓	Write / public	**Passed**
blacklistAddress	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
excludeFromDividends	✓	Write / public	**Passed**
claim	✓	Write / public	**Passed**
claimAddress	✓	Write / public	**Passed**

excludeFromFees	✓	Write / public	**Passed**
excludeMultipleAccountsFromFees	✓	Write / public	**Passed**
processDividendTracker	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setAutomatedMarketMakerPair	✓	Write / public	**Passed**
setLastProcessedIndex	✓	Write / public	**Passed**
setLiquiditFee	✓	Write / public	**Passed**
setMarketingFee	✓	Write / public	**Passed**
setMarketingWallet	✓	Write / public	**Passed**
setSwapTokensAtAmount	✓	Write / public	**Passed**
setUSDTRewardsFee	✓	Write / public	**Passed**
swapOnDemand	✓	Write / public	**Passed**
updateClaimWait	✓	Write / public	**Passed**
updateDividendTracker	✓	Write / public	**Passed**
updateGasForProcessing	✓	Write / public	**Passed**
updateUniswapV2Router	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Front running.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**
18	Design Logic.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found

Medium:

No Medium severity vulnerabilities were found.

Low:

Issue #1.

#Check-effects-interaction:

In detail

Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability.

```
function _withdrawDividendOfUser(address payable user) internal returns
(u uint256 _withdrawableDividend = withdrawableDividendOf(user);
if (_withdrawableDividend > 0) {

withdrawnDividends[user] = withdrawnDividends[user].add(_withdrawable
emit DividendWithdrawn(user, _withdrawableDividend);
(bool success,) = user.call{value: _withdrawableDividend, gas: 3000}(

if(!success) {
withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawab
return 0;

}

     return _withdrawableDividend;}
   return 0;}
```

Issue #2.

#Delete from dynamic array:

```
delete map.inserted[key];
        delete map.values[key];

uint index = map.indexOf[key];
uint lastIndex = map.keys.length - 1; address lastKey =
map.keys[lastIndex];

        map.indexOf[lastKey] = index;
        delete map.indexOf[key];
```

In detail

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

Issue #3.
#Transaction origin:

```
 emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex,
false, gas, tx.origin);
emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex,
true, gas, tx.origin);
```

In detail
Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

Very Low:
Issue #1. For loop over a dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

```
if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
_lastProcessedIndex = 0;

}
```

Issue #2. Low level calls:

In detail Use of "call": should be avoided whenever possible.It can lead to unexpected behavior if return value is not handled properly.

Please use Direct Calls via specifying the called contract's interface.

```
(bool success,) = user.call{value: _withdrawableDividend, gas:
3000}("");

(bool success,) = address(dividendTracker).call{value:
bnbForReflection}("");
```

Notes:

#Note1

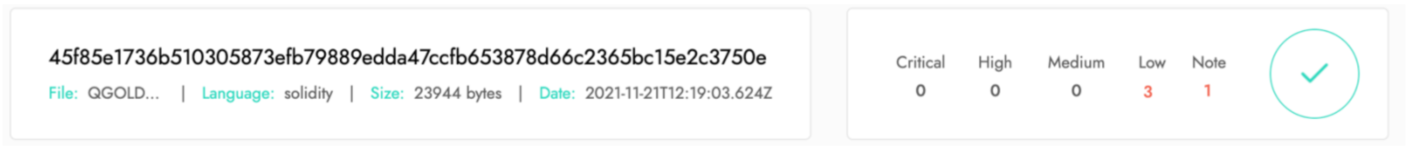
#ERC20:

```
     function decimals() external pure returns
     (uint8);
```

In detail
ERC20 contract's "decimals" function should have "uint8" as return type.

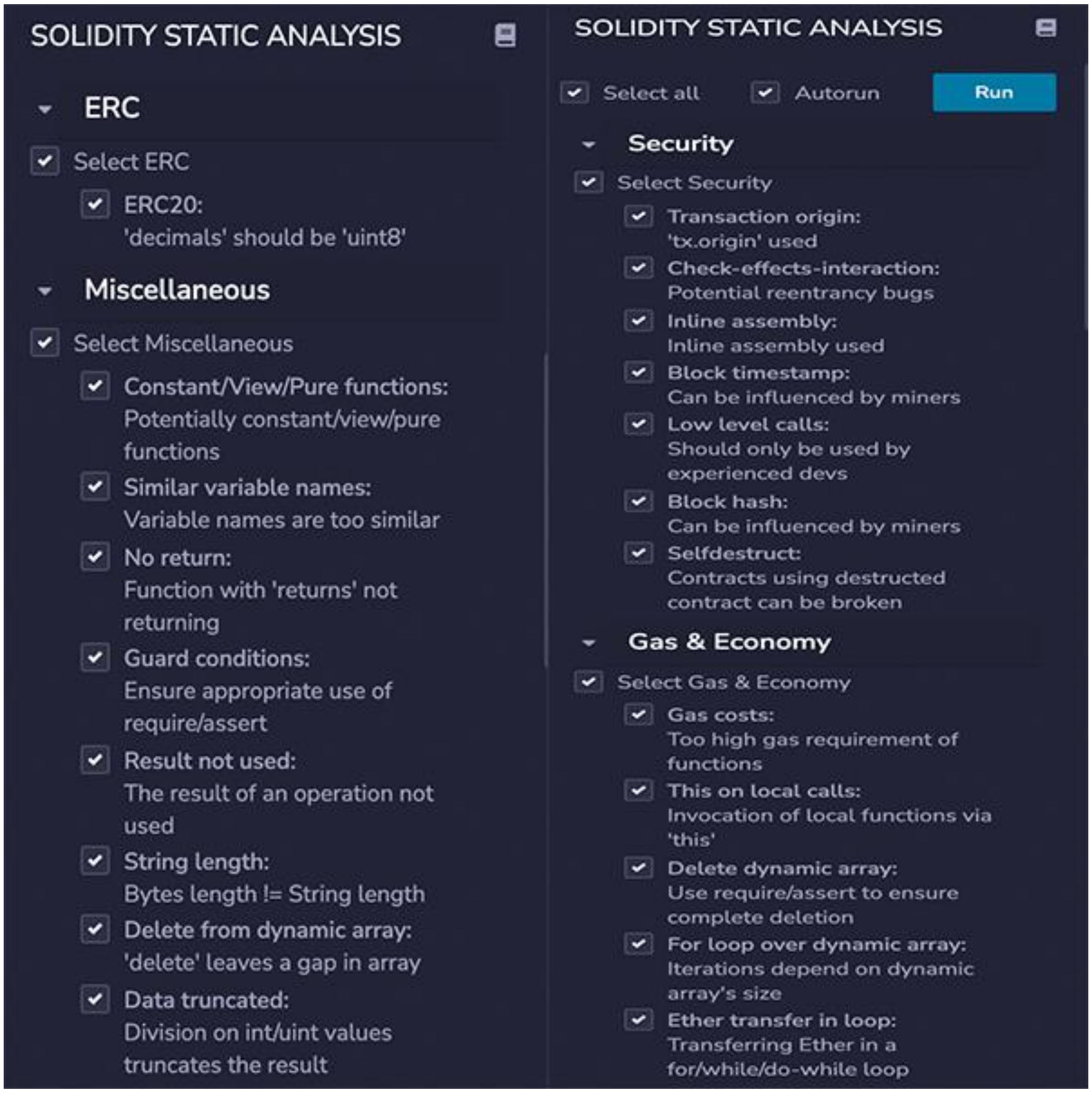
Automatic Testing

1- Check for security


45f85e1736b510305873efb79889edda47ccfb653878d66c2365bc15e2c3750e
File: QGOLD... | Language: solidity | Size: 23944 bytes | Date: 2021-11-21T12:19:03.624Z

Critical	High	Medium	Low	Note
0	0	0	3	1

2- SOLIDITY STATIC ANALYSIS


SOLIDITY STATIC ANALYSIS

ERC

✓ Select ERC
- ✓ ERC20: 'decimals' should be 'uint8'

Miscellaneous

✓ Select Miscellaneous
- ✓ Constant/View/Pure functions: Potentially constant/view/pure functions
- ✓ Similar variable names: Variable names are too similar
- ✓ No return: Function with 'returns' not returning
- ✓ Guard conditions: Ensure appropriate use of require/assert
- ✓ Result not used: The result of an operation not used
- ✓ String length: Bytes length != String length
- ✓ Delete from dynamic array: 'delete' leaves a gap in array
- ✓ Data truncated: Division on int/uint values truncates the result

SOLIDITY STATIC ANALYSIS

✓ Select all ✓ Autorun Run

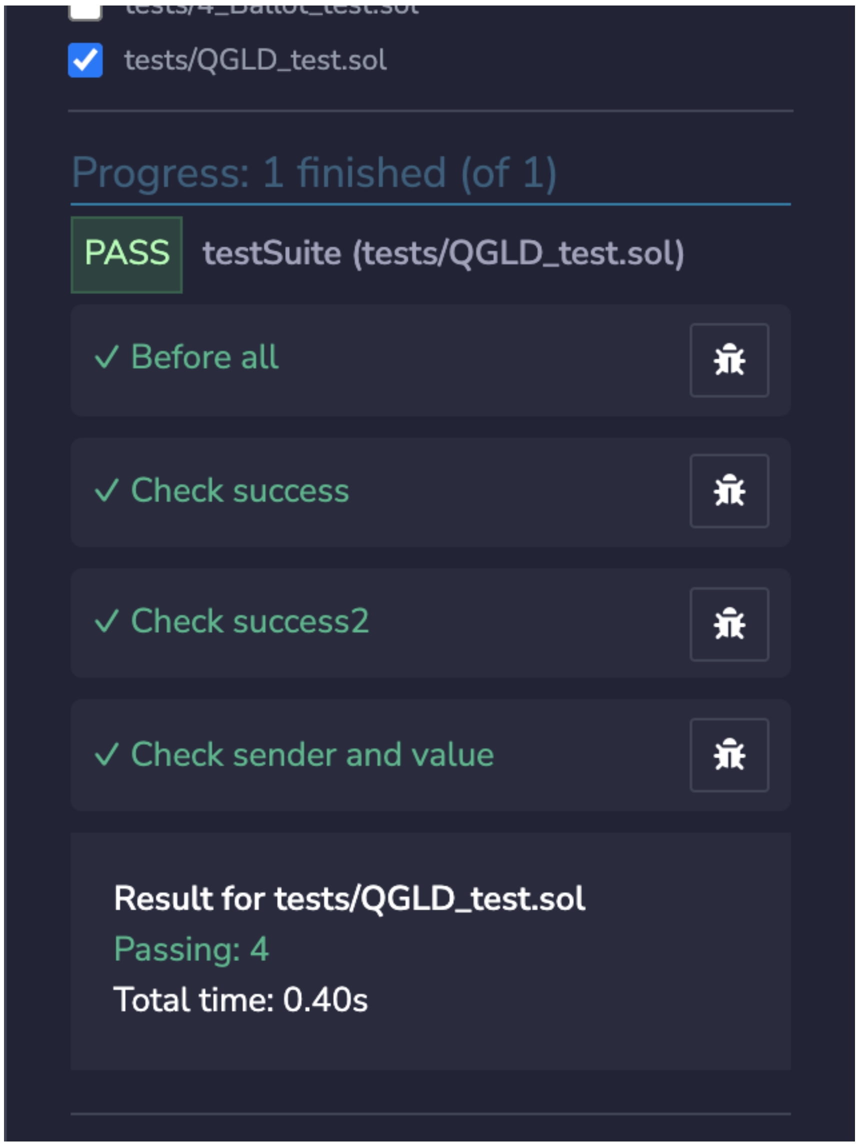
Security

✓ Select Security
- ✓ Transaction origin: 'tx.origin' used
- ✓ Check-effects-interaction: Potential reentrancy bugs
- ✓ Inline assembly: Inline assembly used
- ✓ Block timestamp: Can be influenced by miners
- ✓ Low level calls: Should only be used by experienced devs
- ✓ Block hash: Can be influenced by miners
- ✓ Selfdestruct: Contracts using destructed contract can be broken

Gas & Economy

✓ Select Gas & Economy
- ✓ Gas costs: Too high gas requirement of functions
- ✓ This on local calls: Invocation of local functions via 'this'
- ✓ Delete dynamic array: Use require/assert to ensure complete deletion
- ✓ For loop over dynamic array: Iterations depend on dynamic array's size
- ✓ Ether transfer in loop: Transferring Ether in a for/while/do-while loop

3- SOLIDITY UNIT TESTING



tests/QGLD_test.sol

Progress: 1 finished (of 1)

PASS testSuite (tests/QGLD_test.sol)

✓ Before all

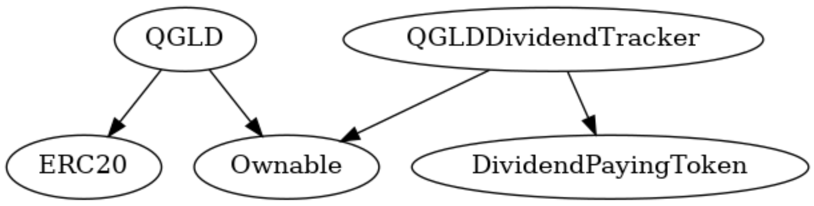
✓ Check success

✓ Check success2

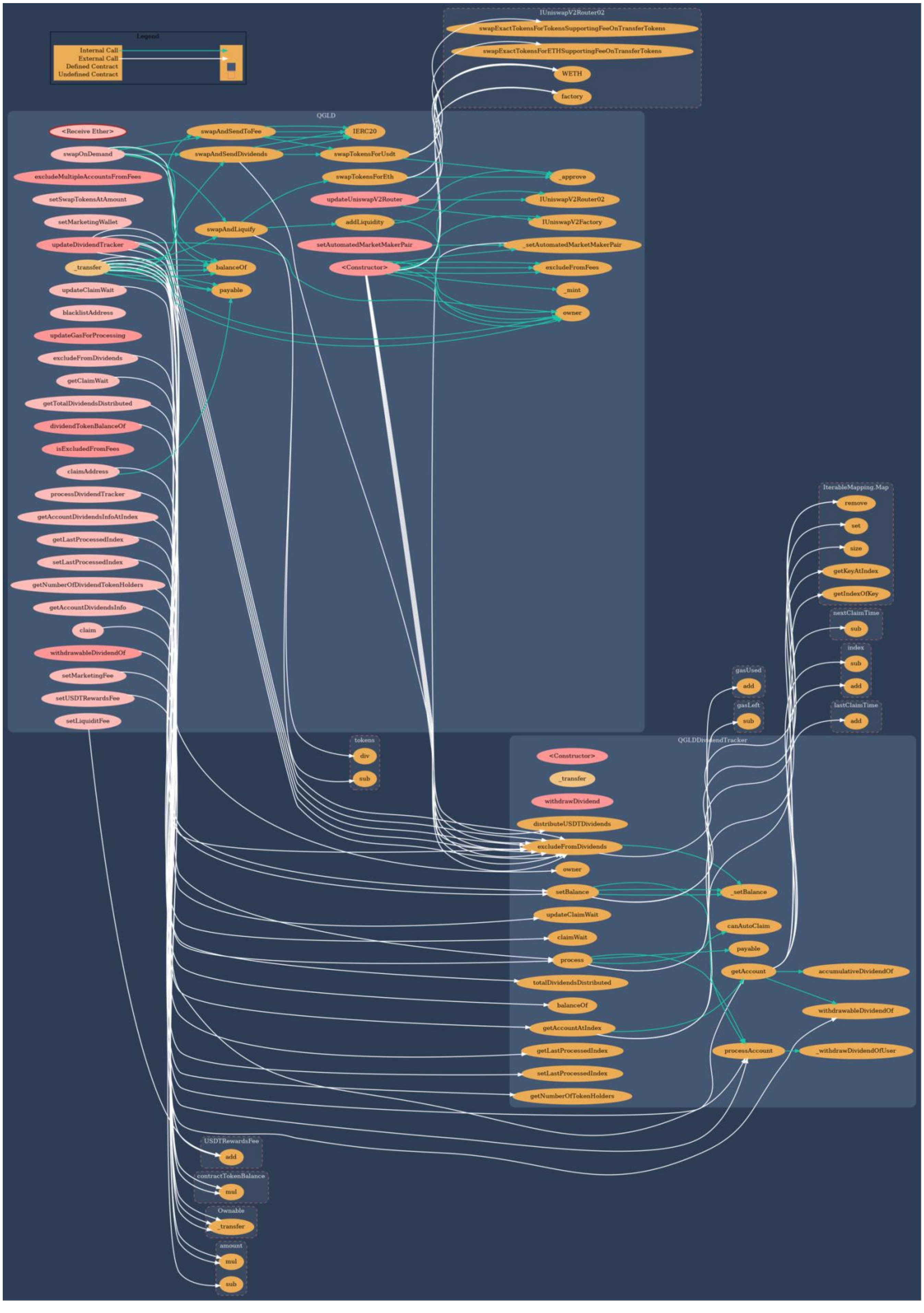
✓ Check sender and value

Result for tests/QGLD_test.sol
Passing: 4
Total time: 0.40s

#Inheritance graph



#Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Router02
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
QGLD
<Receive Ether>
swapOnDemand
excludeMultipleAccountsFromFees
setSwapTokensAtAmount
setMarketingWallet
updateDividendTracker
_transfer
updateClaimWait
blacklistAddress
updateGasForProcessing
excludeFromDividends
getClaimWait
getTotalDividendsDistributed
dividendTokenBalanceOf
isExcludedFromFees
claimAddress
processDividendTracker
getAccountDividendsInfoAtIndex
getLastProcessedIndex
setLastProcessedIndex
getNumberOfDividendTokenHolders
getAccountDividendsInfo
claim
withdrawableDividendOf
setMarketingFee
setUSDTRewardsFee
setLiquiditFee
swapAndSendToFee
swapAndSendDividends
swapAndLiquify
balanceOf
payable
IERC20
swapTokensForUsdt
swapTokensForEth
updateUniswapV2Router
addLiquidity
setAutomatedMarketMakerPair
<Constructor>
_approve
IUniswapV2Router02
IUniswapV2Factory
_setAutomatedMarketMakerPair
excludeFromFees
_mint
owner
IterableMapping.Map
remove
set
size
getKeyAtIndex
getIndexOfKey
nextClaimTime
sub
index
sub
add
gasUsed
add
gasLeft
sub
lastClaimTime
add
tokens
div
sub
QGLDDividendTracker
<Constructor>
_transfer
withdrawDividend
distributeUSDTDividends
excludeFromDividends
owner
setBalance
updateClaimWait
claimWait
process
totalDividendsDistributed
balanceOf
getAccountAtIndex
getLastProcessedIndex
setLastProcessedIndex
getNumberOfTokenHolders
_setBalance
canAutoClaim
payable
getAccount
accumulativeDividendOf
withdrawableDividendOf
processAccount
_withdrawDividendOfUser
USDTRewardsFee
add
contractTokenBalance
mul
Ownable
_transfer
amount
mul
sub

Automatic general report

```
Files Description Table


|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/QGOLD.sol | a4cfcd67248e3b5af813f567c3e9f83aafa1ce06 |


 Contracts Description Table


|  Contract  |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |  **Modifiers**  |
||||||
| **QGLD** | Implementation | ERC20, Ownable |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 |
| └ | <Receive Ether> | External ❗ |  💵 |NO❗ |
| └ | updateDividendTracker | Public ❗ | 🛑  | onlyOwner |
| └ | updateUniswapV2Router | Public ❗ | 🛑  | onlyOwner |
| └ | excludeFromFees | Public ❗ | 🛑  | onlyOwner |
| └ | excludeMultipleAccountsFromFees | Public ❗ | 🛑  | onlyOwner |
| └ | setSwapTokensAtAmount | External ❗ | 🛑  | onlyOwner |
| └ | setMarketingWallet | External ❗ | 🛑  | onlyOwner |
| └ | setUSDTRewardsFee | External ❗ | 🛑  | onlyOwner |
| └ | setLiquiditFee | External ❗ | 🛑  | onlyOwner |
| └ | setMarketingFee | External ❗ | 🛑  | onlyOwner |
| └ | setAutomatedMarketMakerPair | Public ❗ | 🛑  | onlyOwner |
| └ | blacklistAddress | External ❗ | 🛑  | onlyOwner |
| └ | _setAutomatedMarketMakerPair | Private 🔐 | 🛑  | |
| └ | updateGasForProcessing | Public ❗ | 🛑  | onlyOwner |
| └ | updateClaimWait | External ❗ | 🛑  | onlyOwner |
| └ | swapOnDemand | External ❗ | 🛑  | onlyOwner |
| └ | getClaimWait | External ❗ |   |NO❗ |
| └ | getTotalDividendsDistributed | External ❗ |   |NO❗ |
| └ | isExcludedFromFees | Public ❗ |   |NO❗ |
| └ | withdrawableDividendOf | Public ❗ |   |NO❗ |
| └ | dividendTokenBalanceOf | Public ❗ |   |NO❗ |
| └ | excludeFromDividends | External ❗ | 🛑  | onlyOwner |
| └ | getAccountDividendsInfo | External ❗ |   |NO❗ |
| └ | getAccountDividendsInfoAtIndex | External ❗ |   |NO❗ |
| └ | processDividendTracker | External ❗ | 🛑  |NO❗ |
| └ | claim | External ❗ | 🛑  |NO❗ |
| └ | claimAddress | External ❗ | 🛑  | onlyOwner |
| └ | getLastProcessedIndex | External ❗ |   |NO❗ |
| └ | setLastProcessedIndex | External ❗ | 🛑  | onlyOwner |
| └ | getNumberOfDividendTokenHolders | External ❗ |   |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | swapAndSendToFee | Private 🔐 | 🛑  | |
| └ | swapAndLiquify | Private 🔐 | 🛑  | |
| └ | swapTokensForEth | Private 🔐 | 🛑  | |
| └ | swapTokensForUsdt | Private 🔐 | 🛑  | |
| └ | addLiquidity | Private 🔐 | 🛑  | |
| └ | swapAndSendDividends | Private 🔐 | 🛑  | |
||||||
| **QGLDDividendTracker** | Implementation | Ownable, DividendPayingToken |||
| └ | <Constructor> | Public ❗ | 🛑  | DividendPayingToken |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | withdrawDividend | Public ❗ | 🛑  |NO❗ |
| └ | excludeFromDividends | External ❗ | 🛑  | onlyOwner |
| └ | updateClaimWait | External ❗ | 🛑  | onlyOwner |
| └ | setLastProcessedIndex | External ❗ | 🛑  | onlyOwner |
| └ | getLastProcessedIndex | External ❗ |   |NO❗ |
| └ | getNumberOfTokenHolders | External ❗ |   |NO❗ |
| └ | getAccount | Public ❗ |   |NO❗ |
| └ | getAccountAtIndex | Public ❗ |   |NO❗ |
| └ | canAutoClaim | Private 🔐 |   | |
| └ | setBalance | External ❗ | 🛑  | onlyOwner |
| └ | process | Public ❗ | 🛑  |NO❗ |
| └ | processAccount | Public ❗ | 🛑  | onlyOwner |


 Legend


|  Symbol  |  Meaning  |
|:--------:|-----------|
|    🛑    | Function can modify state |
|    💵    | Function is payable |
```

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “secured”.

- ✓ No mint function.
- ✓ No volatile code.
- ✓ Not many high severity issues were found.
- ✓ Contract Ownership Renounced.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn’t say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.