DEXMON

Smart Contract Security Audit

Questos

Jul 25th, 2021

Table of Contents

Summary

This report has been made for **Questos** to discover issues and vulnerabilities in the source code of the smart contract. Automatic code Analysis has been performed as well as a manual code review.

The audit process has given special attention to:

- Check the code against attack vectors.
- Ensure compliance with current industry standards.
- Ensure contract logic meets the specifications of the project.
- Compare the contract to other contract implementations industry leaders.

This security report resulted in different types of findings from medium to info. We recommend addressing the findings, if possible, to ensure a high security level and industry best practices.

Overview

Project Summary

Project name:	Questos
Description:	Questos is carbon negative blockchain. Our project absorbs more harmful CO2 than it causes.
Plattform:	BSC
Language:	Solidity
Codebase:	Questos.sol

Audit Summary

Date:	2021/07/24
Plattform:	Binance Smart Chain
Audit Methodology:	Automated Tests, Manual Review, Testnet Deployment
Key Components:	Questos.sol

Contract Summary

Total Supply:	1.000.000.000
Decimals:	18
Platform:	Binance Smart Chain
Compiler version:	v0.6.12+commit.27d51765
Symbol:	xQTX
Name:	Questos - Green Blockchain

Vulnerability Summary

Total Issues	14
Critical	0
Major	0
Medium	3
Minor	8
Info	3

Audit Scope

File	SHA256
Questos.sol	5fc7583cd4c3bbc5eb437b312e40bf675af431cf0a5ca56697ad48292fcc0651

Contract Overview

Contract Description

The Questos Protocol is a decentralized finance token deployed on the Binance Smart Chain (BSC) network. Questos uses two advanced features in its protocol. Static rewards for the holders of Questos Token and an automated liquidity function.

The static reward (reflection) is implemented through a 5% transfer fee and is distributed to all token holders.

The automated liquidity function (auto LP) is implemented through a 5% transfer fee and is added to the PancakeSwapv2 liquidity pool.

Additionally, there is a 1% charity, advertising and development fee that will be send to an extra wallet. There is a 1% Burn fee.

Contract Functions

Privileged Functions

The contract includes the following privileged functions, which are restricted by the "`onlyOwner`" modifier and used to change the contract specifications and address attributes.

Change in the scheme's liquidation, tax and maximum transaction percentages:

- `function setTaxFeePercent(uint taxFee)`
- `function setMaxTxPercent(uint maxTxPercent)`
- `function setMaxTxPercent(uint maxTxPercent)`
- `function enableAllFees()`
- `function disableAllFee()`

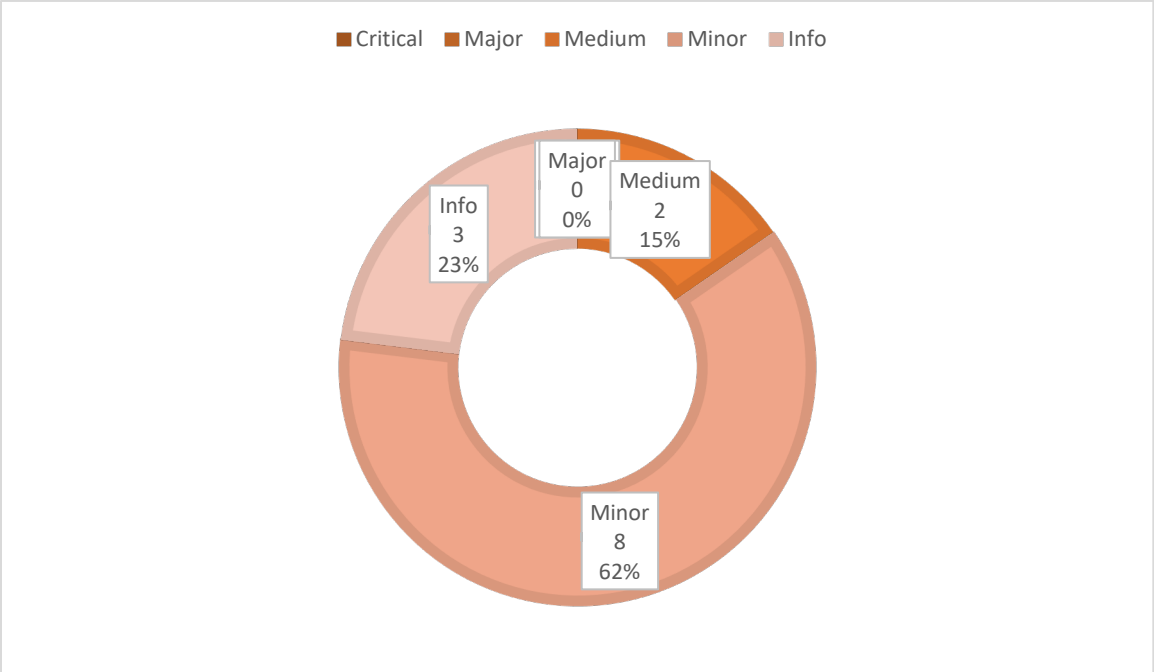
Account management functions for the reward system:

- `function excludeFromReward(address account)`
- `function includeInReward(address account)`
- `function excludeFromFee(address account)`
- `function includeInFee(address account)`

Function to toggle the auto LP mechanism:

- `function setSwapAndLiquifyEnabled(bool _enabled)`

Findings



Summary

No	Type	Description
#Sec-01	Minor	private _tTotal
#Sec-02	Minor	private _decimals
#Sec-03	Minor	private _name
#Sec-04	Minor	private _symbol
#Sec-05	Minor	private numTokensSellToAddToLiquidity
#Sec-06	Minor	Wrong error message in code
#Sec-07	Medium	Contract gains BNB that is not withdrawable
#Sec-08	Info	Return value not handled

#Sec-09	Info	Naming is not matching the operating environment
#Sec-10	Minor	3rd party dependencies
#Sec-11	Minor	Privileged owners
#Sec-12	Info	Typos in the contract
#Sec-13	Medium	Possible to gain ownership after renouncing
#Sec-14	Medium	Centralized risk in addLiquidity

#Sec-01: private _tTotal

Type: **Minor**

```
    uint256 private _tTotal = 1000000000 * 10 ** 9;
```

Description

Variable `private _tTotal` can be declared as `constant` since the variable is never changed.

Recommendation

We recommend defining this variable as `constant`.

#Sec-02: private _decimals

Type: **Minor**

```
    uint8 private _decimals = 9;
```

Description

Variable `private _decimals` can be declared as `constant` since the variable is never changed.

Recommendation

We recommend defining this variable as `constant`.

#Sec-03: private _name

Type: **Minor**

```
string private _name = "NinjaDoge";
```

Description

Variable `private _name` can be declared as `constant` since the variable is never changed.

Recommendation

We recommend defining this variable as `constant`.

#Sec-04: private _symbol

Type: **Minor**

```
    string private _symbol = "NINJADOGE";
```

Description

Variable `private _symbol` can be declared as `constant` since the variable is never changed.

Recommendation

We recommend defining this variable as `constant`.

#Sec-05: private numTokensSellToAddToLiquidity

Type: **Minor**

```
uint256 private numTokensSellToAddToLiquidity = 500000 * 10 ** 9;
```

Description

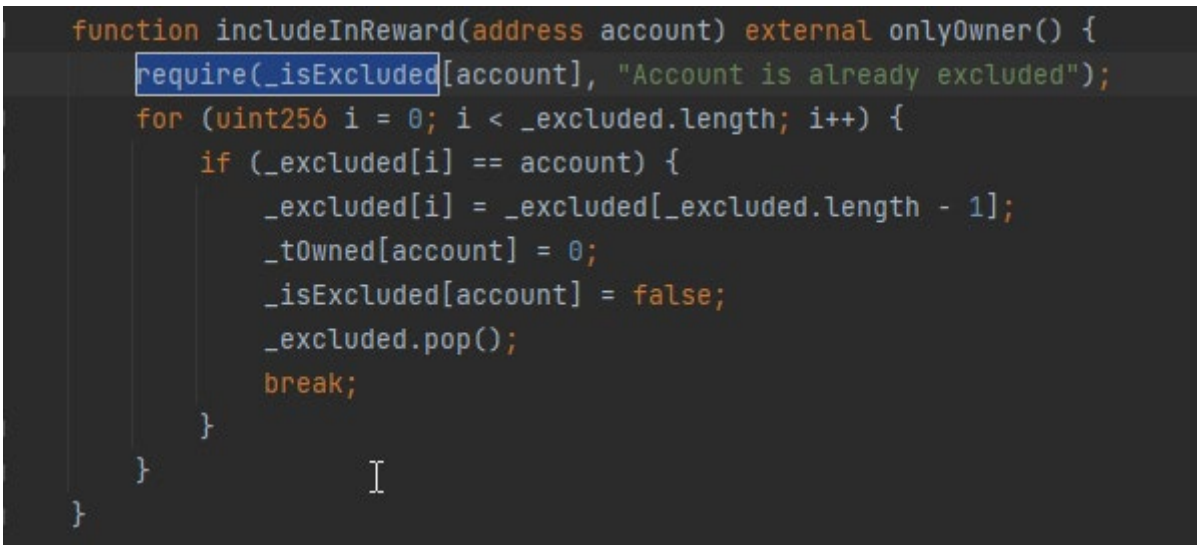
Variable `private numTokensSellToAddToLiquidity` can be declared as `constant` since the variable is never changed.

Recommendation

We recommend defining this variable as `constant`.

#Sec-06: Wrong error message in Code

Type: **Minor**



```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

Description

In the function `includeInReward` the message `"Account is already excluded"` is does not describe the error correctly.

Recommendation

The message `"Account is already excluded"` should be changed to `"Account not excluded"`.

#Sec-07: Contract gains BNB that is not withdrawable

Type: **Medium**

```
function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
    // split the contract balance into halves
    uint256 half = contractTokenBalance.div(2);
    uint256 otherHalf = contractTokenBalance.sub(half);

    // capture the contract's current ETH balance.
    // this is so that we can capture exactly the amount of ETH that the
    // swap creates, and not make the liquidity event include any ETH that
    // has been manually sent to the contract
    uint256 initialBalance = address(this).balance;

    // swap tokens for ETH
    swapTokensForEth(half);
    // <- this breaks the ETH -> HATE swap when swap+liquify is triggered

    // how much ETH did we just swap into?
    uint256 newBalance = address(this).balance.sub(initialBalance);

    // add liquidity to uniswap
    addLiquidity(otherHalf, newBalance);

    emit SwapAndLiquify(half, newBalance, otherHalf);
}
```

Description

The function swapAndLiquify 50% of the contractTokenBalance Questos tokens into BNB. The other half of the Questos tokens and part of the converted BNB are paid into the Questos BNB pancake swap liquidity pool.

Each time the function swapAndLiquify is called, a small amount of BNB remains in the contract, because the Questos price drops a bit after the first half of Questos tokens are swapped to BNB and the other half of Questos need less than the converted BNB to be paired with it when liquidity is added. The contract does not have a way to withdraw these BNB. They will be locked into the contract forever.

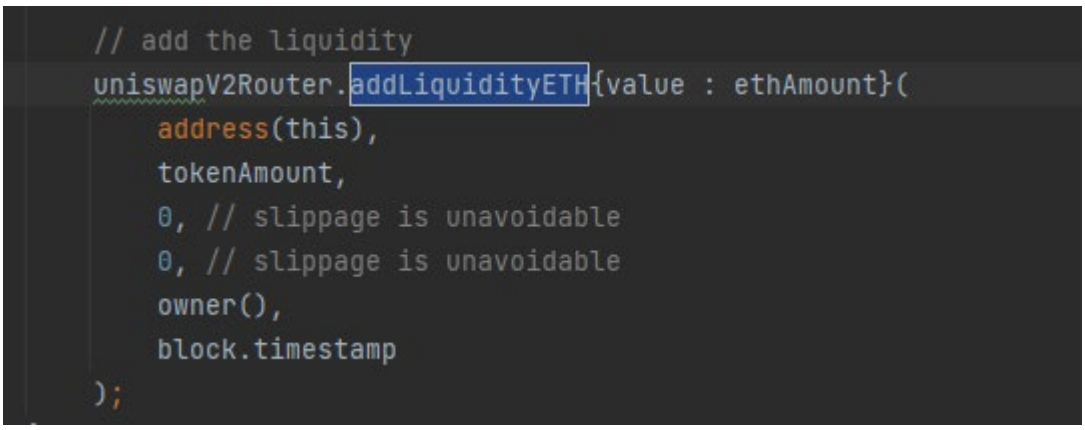
Recommendation

This is not ideal that more and more BNB gets locked into the contract over time.

One solution could be to add a function into the contract that can withdraw the BNB.

#Sec-08: Return value not handled

Type: **Info**



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

Description

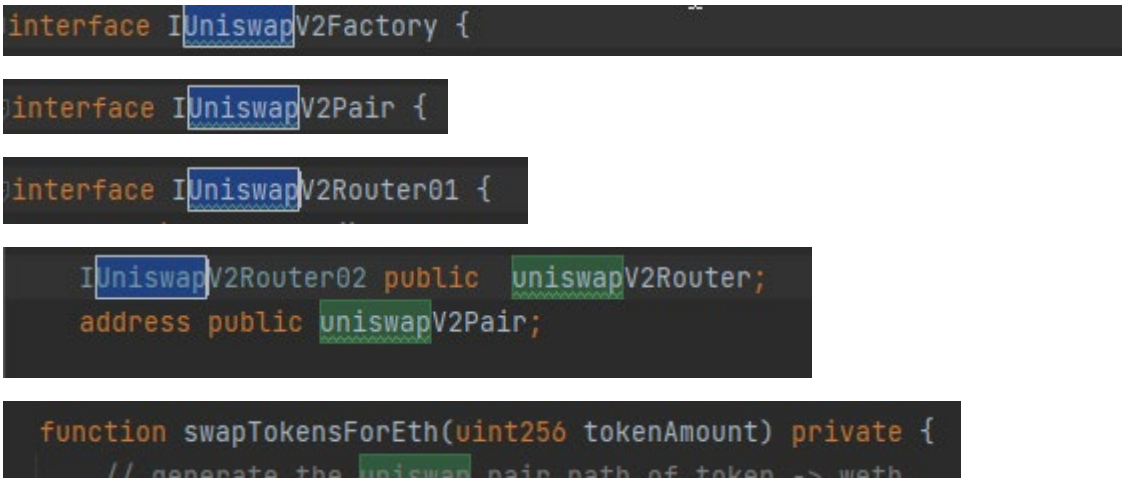
The functions return value is not properly handled.

Recommendation

We recommend using variables to receive the return value.

#Sec-09: Naming is not matching the operating environment

Type: **Info**



Description

The Questos Contract uses the Binance Smart Chain network and PancakeSwapv2. In the contract the naming is Uniswap and ETH.

Recommendation

We recommend changing “Uniswap” and “ETH” to “Pancakeswap” and “BNB”.

#Sec-10: 3rd party dependency

Type: **Minor**

Description

The Questos Contract depends on the PancakeSwap protocols. This audits scope was only the Questos contract. We assume that 3rd party dependencies function correctly. However, there is always a very small risk that 3rd party dependencies can be compromised or changed.

Recommendation

The interaction with PancakeSwap protocols is needed in the logic of the auto liquidity function of the Questos protocol. Therefore, we recommend the Questos team to monitor the 3rd party dependencies and deactivate the auto liquidity feature when unexpected activities are observed at the 3rd party side.

#Sec-11: Ownership privileged

Type: **Minor**

Description

The Questos Contract Owner has the permission to:

I. change taxFee, liqidityFee, _maxTxAmount and setCharityWallet
II. exclude and include addresses from rewards.
III. Enable and disable auto LP function.
IV. Change the LP token receive address.

Recommendation

Renounce ownership or time lock the ownership.

#Sec-12: Ownership privileged

Type: **Info**

Description

```
event SwapAndLiquify(
    uint256 tokensSwapped,
    uint256 ethReceived,
    uint256 tokensIntoLiqudity
);
```

tokensIntoLiqudity should be named to tokensIntoLiquidity.

```
//to recieve ETH from uniswapV2Router when swaping
receive() external payable {}
```

recieve should be receive and swaping should be swapping.

Recommendation

We recommend correcting the typos.

#Sec-13: Possible to gain ownership after renouncing

Type: **Medium**

Description

It is possible for an owner to gain ownership after call of the renounceOwnership function. This can be done by performing following actions:

1. Call lock. (_previousOwner will be set to current owner.)
2. Call unlock.
3. Call renounceOwnership.
4. Call unlock to get ownership again.

Recommendation

We recommend to remove unlock and lock functions or change the code in the renounceOwnership function to set _previousOwner to 0 address.

Disclamer

The following terminology applies to these Terms and Conditions, Privacy Statement and Disclaimer Notice and all Agreements: "Client", "You" and "Your" refers to you, the person log on this report and compliant to the Company's terms and conditions. "The Company", "Ourselves", "We", "Our" and "Us", refers to our Company. "Party", "Parties", or "Us", refers to both the Client and ourselves.

Unless otherwise stated, Dexmon and/or its licensors own the intellectual property rights for all material on Dexmon. All intellectual property rights are reserved. You may access this from Dexmon for your own personal use subjected to restrictions set in these terms and conditions.

This report is subject of the terms and conditions of Dexmon. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.
The Dexmon team only audited the smart contract by their best knowing at with the technology at the time the report was made.

Contact

cert@dexmon.com

dexmon.com