[**XoloInu**](https://inu-xolo.gitbook.io/xolo-inu/) **Smart Contract Initial Audit Report**

[**Overview**](#_a3yxug87siiq) **2**

[Scope of Audit](#_zf94eajpvmc3) 2

[Check Vulnerabilities](#_fk11hakqdckb) 2

[**Techniques and Methods**](#_kmnjskexwpfd) **3**

[Issue Categories](#_2p6a8gpqk15r) 4

[Number of security issues per severity.](#_2idbboidjlj7) 5

[**Introduction**](#_41paf4h67ss0) **5**

[**Issues Found – Code Review / Manual Testing**](#_jnyy7e310pds) **6**

[High Severity Issues](#_fpa7vs16wijg) 6

H.1 Contract code size exceeds 24576 bytes

[Medium Severity Issues](#_ihaplnhuvava) 6

[Low Severity Issues](#_3xubtpht259r) 6

L[.1](#_xg8p62iavmvd) Use of block.timestamp 6

L[.2](#_rxz9j7wxz3i1) Missing Zero Address Validation 7

[Informational Issues](#_urpasvdhsbj6) 7

INF[.1](#_bnfy0uvce8ul)Missing comments and description 7

INF[.](#_bnfy0uvce8ul)2 Incorrect and Inconsistent use of Indentations and Spaces 7

INF[.](#_bnfy0uvce8ul)3Less meaningful variable and method names 8

INF[.](#_bnfy0uvce8ul)4Incorrect method names 8

INF.5 Variables defined but never used 8

INF.6 Public methods only being used externally 9

INF.7 Constant calculations in the contract 9

INF.8 Use constructor to set addresses 10

INF.9 Reading address(this) multiple times 10

INF.10 Getters should be at the bottom of the contract 11

INF.11 Length calculation within the loop 11

INF.12 Ordering conditions correctly can reduce boolean operations 11

INF.13 Inconsistent error messages 12

INF.14 Keeping the rarest condition check first 12

INF.15 Redundant calculations 13

INF.16Inefficient Strict Comparisons 13

INF.17 Same execution for two different cases can be merged 14

INF.18 Avoid hardcoded addresses 15

INF.19 Unnecessary use of SafeMath 15

INF.20 Misleading Comment 15

[**Closing Summary**](#_wt0t7ihkiqxh) **16**

[**Disclaimer**](#_i7nr8bi7b69r) **16**

## **Overview**

## **Scope of Audit**

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

## **Check Vulnerabilities**

* 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 contracts, 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 analyzed 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**

A 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 analyzed, 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.

### 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.

**High Severity Issues**

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 Severity Issues**

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 Severity Issues**

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** **Issues**

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 security issues per severity.**

## 

| **TYPE** | **HIGH** | **MEDIUM** | **LOW** | **INFORMATIONAL** |
| --- | --- | --- | --- | --- |
| **Open** | **1** | **0** | **2** | **20** |
| **Acknowledged** | **0** | **0** | **0** | **0** |
| **Closed** | **0** | **0** | **0** | **0** |

# **Introduction**

During the period of **November 13, 2021 to November 14, 2021** - QuillAudits Team performed a security audit for **XoloInu** smart contracts.

The code for the audit was taken from following the official link**:**

**Codebase:** [**https://github.com/dopa-admin/Xolo-Inu**](https://github.com/dopa-admin/Xolo-Inu)

| **Version Number** | **Date** | **Commit ID** | **Branch** |
| --- | --- | --- | --- |
| 01 | October | 1732ddc19ff59ad9f4ed38c13236862be0b7b037 | Master |

**Contract - XoloInu**

# **Issues Found – Code Review / Manual Testing**

## **High Severity Issues**

### [H1] Contract code size exceeds 24576 bytes

Contract implementation is too large in size to be deployed on mainnet.  
Ethereum with its spurious dragon release limited the size of the contracts deployable on mainnet to 24576 bytes.

The size of the contract XoloInu.sol goes way above this value and currently is of size **28055** bytes

**Recommendation**: Define and use libraries for pure and view functions e.g.  
 We can create a library which contains all the mathematical operations.

**Status**: **Open**

## **Medium Severity Issues**

**None**

## **Low Severity Issues**

### [L1] Use of block.timestamp for comparisons

The value of block.timestamp can be manipulated by the miner.

And conditions with strict equality is difficult to achieve -

**block.timestamp == launchTime**

**Recommendation**: Avoid use of block.timestamp

**Status**: **Open**

### [L2] Missing Zero Address Validation

Function **\_transfer()**: Missing Zero Address Check for **to** address

**Recommendation**: Add a ‘require’ to check **to** address != address(0)

**Status**: **Open**

## **Informational Issues**

### [INF1] Missing comments and description:

Comments and Description of the methods and the variables are missing, it's hard to read and understand the purpose of the variables and the methods in context of the whole picture

**Recommendation**: Consider adding NatSpec format comments for the  
 comments and state variables

**Status**: **Open**

### [INF2] Incorrect and Inconsistent use of Indentations and Spaces:

Throughout the contract there are several lines which have not been indented properly in consistency with the other lines of code.  
There are several extra spaces present in the empty lines and also added after and in between the characters in a particular line

**Recommendation**: Trim all the extra spaces and indent all the lines correctly

**Status**: **Open**

### [INF3] Less meaningful variable and method names:

Certain variables and method names does not provide clear picture of their purpose  
  
**Recommendation:**

* **MAX** should be renamed to **MAX\_INT\_256**
* **'inSwapAndLiquify'** can be simply named **'\_locked'**
* **"newBalance"** in **swapAndLiquify** should be called **"addedBalance"** or **"swappedEth"**
* **"takeMarketing"** should be called **"takeMarketingFee"**

**Status**: **Open**

### [INF4] Incorrect method names

Incorrect names used on method caused confusion while usage and can be problematic in the future.  
Function **isRemovedSniper** returns whether the passed address isSniper or not.

Function **\_removeSniper** is adding setting isSniper to for the passed address

**Recommendation:**

* **isRemovedSniper** should be renamed to **isSniper**
* **\_removeSniper** should be renamed to **addSniper**

**Status**: **Open**

### [INF5] Variable defined but never used

**'deadAddress'** defined but never used

**Recommendation:** Remove unused variables

**Status**: **Open**

### [INF6] Public methods only being used externally

‘public’ functions that are never by the contract should be declared ‘external’  
 to save gas.

**Recommendation:** Make these methods external - **setSwapAndLiquifyEnabled, setRouterAddress, isRemovedSniper, includeInFee, excludeFromFee, isExcludedFromFee, excludeFromReward, reflectionFromToken, deliver, totalFees, isExcludedFromReward**

**Status**: **Open**

### [INF7] Constant calculations in the contract

Precalculated initialization will save 2847 units of gas in deployment



**Recommendation:** Replace the initialization as -

uint256 private \_tTotal = 10000000000000000000000000000; // 10000000000 \*  
 10\*\*18; // 10000000000 \* 10\*\*18;

uint256 private \_rTotal = 115792089237316195423570985008687907853269984665640000000000000000000000000000 // (MAX - (MAX % \_tTotal));

10\*\*2 should be a precalculated constant and then used in **calculateTaxFee**   
 and **calculateLiquidityFee** to save more than 10 units on gas on every call

**Status: Open**

### [INF8] Use constructor to set addresses

Address being set in contract should be provided dynamically via constructor on deployment for readability and correctness

**Recommendation:** Pass the address of **“marketingWallet”** and **“uniswapV2Router”** as a parameter of the constructor and set the value

**Status: Open**

### [INF9] Reading address(this) multiple times

Methods **constructor, \_takeLiquidity, swapAndLiquify, swapTokensForEth,**

**addLiquidity and takeMarketing** are reading and typecasting several times in the same block costing extra gas.

**Recommendation:** Define a local address variable and use it instead to avoid repeated type casting

**Status: Open**

### [INF10] Getters should be at the bottom of the contract

**Status: Open**

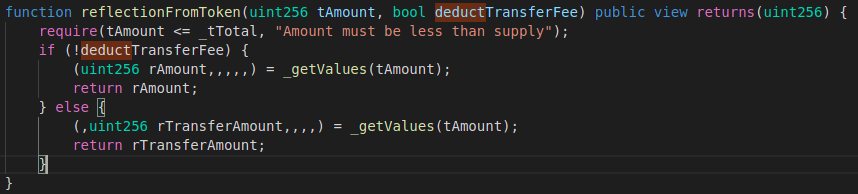
### [INF11] Length calculation within the loop

Reading from the state and fetching its length is a costly operation and doing such within a loop should be avoided

**Recommendation: \_excluded.length** should be calculated and stored in a variable before using in the for loop inside **`\_getCurrentSupply`** method

**Status: Open**

### [INF12] Ordering conditions correctly can reduce boolean operations



**Recommendation:** By recording the if condition we can reduce one boolean operation and increase readability

if (deductTransferFee) {

(,uint256 rTransferAmount,,,,) = \_getValues(tAmount);

return rTransferAmount;

} else {

(uint256 rAmount,,,,,) = \_getValues(tAmount);

return rAmount;

}

**Status: Open**

### [INF13] Inconsistent error messages

The require checks in the contract have error messages containing the contract name in some places and the same has been skipped in most cases.

**Recommendation:** The error message should follow the same pattern and we recommend using the contract name along with the error message at all places  
E.g. require(!\_isExcluded[account], "XoloInu: Account is already excluded");

**Status: Open**

### [INF14] Keeping the rarest condition check first saves gas in cases when it is followed by other condition checks

**// buy**

**if(from == uniswapV2Pair && to != address(uniswapV2Router) && !\_isExcludedFromFee[to]) {**

**require(tradingOpen, "Trading not yet enabled.");**

**//antibot**

**if (block.timestamp == launchTime) {**

**\_isSniper[to] = true;**

**\_confirmedSnipers.push(to);**

**}**

**}**

**Recommendation:**

require(tradingOpen, "Trading not yet enabled.");

if(block.timestamp == launchTime && from == uniswapV2Pair && to != address(uniswapV2Router) && !\_isExcludedFromFee[to]) {

\_isSniper[to] = true;

\_confirmedSnipers.push(to);

}

**Status: Open**

### [INF15] Redundant calculations

// split the Liquidity tokens balance into halves

uint256 half = tokensForLiquidity.div(2);

uint256 otherHalf = tokensForLiquidity.sub(half);

Here both the variables will always have the same value but we are calculating the same value twice and using it in two different variables as well.

**Recommendation:**

uint256 half = tokensForLiquidity.div(2);

uint256 otherHalf = half;

We can also avoid creating another variable and use the variable **half** twice

**Status: Open**

### [INF16] Naming Conventions

The contract follows a consistent naming convention where we are private variables with leading “\_” and public variables without it.

But we have missed to comply to the condition for certain variable names -

**"transferToAddressETH"** and **"takeMarketing"** which are private and

**“\_removeSniper”** which is external

**Recommendation:**

Remove “\_” from external variable names and add it to private variable names

**Status: Open**

### [INF17] Same execution for two different cases can be merged

if (\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferFromExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferToExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferStandard(sender, recipient, amount);

} else if (\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferBothExcluded(sender, recipient, amount);

} else {

\_transferStandard(sender, recipient, amount);

}

**Recommendation:**

if (\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferFromExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferToExcluded(sender, recipient, amount);

} else if (\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferBothExcluded(sender, recipient, amount);

} else {

\_transferStandard(sender, recipient, amount);

}

**\_transferStandard** has been moved to else block and one if case is reduced

**Status: Open**

### [INF18] Avoid hardcoded addresses

require(account != 0x10ED43C718714eb63d5aA57B78B54704E256024E, 'We can not blacklist Uniswap');

**Recommendation:**

use already defined uniswapV2Router for comparison

require(account != address(uniswapV2Router), 'We can not blacklist Uniswap');

**Status: Open**

### [INF19] Unnecessary use of SafeMath

Solidity version 0.8 was released with safeMath checks inbuilt, we can avoid

using an explicit safe math library

**Status: Open**

### [INF20] Misleading comment

“**\_tokenTransfer**” has a misleading comment about a variable called “takeFee”

while there is no such variable defined in the contract

**Status: Open**

# **Closing Summary**

Overall, smart contracts are very well written, documented and adhere to guidelines. Several issues of High and Low severity have been reported.  
The **contract can’t be deployed to public EVM chains** due to the [**EIP150**](https://eips.ethereum.org/EIPS/eip-150) implementation and every contract should have the size below **24576** bytes, we recommend to reduce the contract size and work on suggestions that are reported in order to improve the code quality of contracts.

## **Disclaimer**

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **XoloInu 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 XoloInu Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.