



Cyberscope

Audit Report

Xmas Grok

December 2023

Network BSC

Address 0x5E58EFC56e4081c57b0e43a251a896d3dC5f5310

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	FAI	Function Argument Inconsistency	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	MEM	Misleading Error Messages	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	4
Audit Updates	4
Source Files	4
Findings Breakdown	5
ST - Stops Transactions	6
Description	6
Recommendation	6
DDP - Decimal Division Precision	7
Description	7
Recommendation	7
FAI - Function Argument Inconsistency	8
Description	8
Recommendation	8
PTRP - Potential Transfer Revert Propagation	9
Description	9
Recommendation	9
RSML - Redundant SafeMath Library	10
Description	10
Recommendation	10
MEM - Misleading Error Messages	11
Description	11
Recommendation	11
IDI - Immutable Declaration Improvement	12
Description	12
Recommendation	12
L04 - Conformance to Solidity Naming Conventions	13
Description	13
Recommendation	14
L05 - Unused State Variable	15
Description	15
Recommendation	15
L16 - Validate Variable Setters	16
Description	16
Recommendation	16
L19 - Stable Compiler Version	17
Description	17

Recommendation	17
L20 - Succeeded Transfer Check	18
Description	18
Recommendation	18
Functions Analysis	19
Inheritance Graph	23
Flow Graph	24
Summary	25
Disclaimer	26
About Cyberscope	27

Review

Contract Name	XMasGrok
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Explorer	https://bscscan.com/address/0x5e58efc56e4081c57b0e43a251a896d3dc5f5310
Address	0x5e58efc56e4081c57b0e43a251a896d3dc5f5310
Network	BSC
Symbol	XMGROK
Decimals	9
Total Supply	10,000,000,000

Audit Updates

Initial Audit	19 Dec 2023
---------------	-------------

Source Files

Filename	SHA256
XMasGrok.sol	18c8fe0501f8db1428508626005ad3718ea3320912682f62426f068c985bd2eb

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	11

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	11	0	0	0

ST - Stops Transactions

Criticality	Critical
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner, as described in detail in section [PTRP](#). As a result, the contract may operate as a honeypot.

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	XMasGrok.sol#L304
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

The `amount` variable might not be splitted as expected

```
function sendETHToFee(uint256 amount) private {  
    _developmentAddress.transfer(amount.div(2));  
    _marketingAddress.transfer(amount.div(2));  
}
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

FAI - Function Argument Inconsistency

Criticality	Minor / Informative
Location	XMasGrok.sol#L355,368
Status	Unresolved

Description

The business logic seems peculiar. The implementation may not follow the expected behavior. The arguments on the `_getValue` function are called in the wrong order. As a result, the tax fee is used as a liquidity fee and vice versa.

```
function _getValues(uint256 tAmount) private view returns
(uint256, uint256, uint256, uint256, uint256, uint256) {
    (uint256 tTransferAmount, uint256 tFee, uint256 tTeam) =
    _getTValues(tAmount, _redisFee, _taxFee);
    ...
}
function _getTValues(uint256 tAmount, uint256 taxFee, uint256
TeamFee) private pure returns (uint256, uint256, uint256) {
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	XMasGrok.sol#L304
Status	Unresolved

Description

The contract sends funds to a `_marketingAddress` and `_developmentAddress` as part of the transfer flow. These addresses can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
function sendETHToFee(uint256 amount) private {  
    _developmentAddress.transfer(amount.div(2));  
    _marketingAddress.transfer(amount.div(2));  
}
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	XMasGrok.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	XMasGrok.sol#L386,392
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(_msgSender() == _developmentAddress || _msgSender() ==  
_marketingAddress || _msgSender() == owner())
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	XMasGrok.sol#L166,167
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
uniswapV2Router
uniswapV2Pair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	XMasGrok.sol#L44,142,155,156,157,309,310,315,322,408
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
uint256 private constant _tTotal = 10000 * 10**6 * 10**9
string private constant _name = "XMas Grok"
string private constant _symbol = "XMGROK"
uint8 private constant _decimals = 9
event tokensRescued(address indexed token, address indexed to,
uint amount);
address _tokenAddr
address _to
uint _amount
event devAddressUpdated(address indexed previous, address
indexed adr);
event marketingAddressUpdated(address indexed previous, address
indexed adr);
bool _swapEnabled
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	XMasGrok.sol#L141
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
mapping (address => uint256) private _tOwned
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	XMasGrok.sol#L132,322,329
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_owner = newOwner  
_developmentAddress = dev  
_marketingAddress = markt
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	XMasGrok.sol#L17
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.9;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	XMasGrok.sol#L316
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
Token(_tokenAddr).transfer(_to, _amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the [Openzeppelin library](#).

Functions Analysis

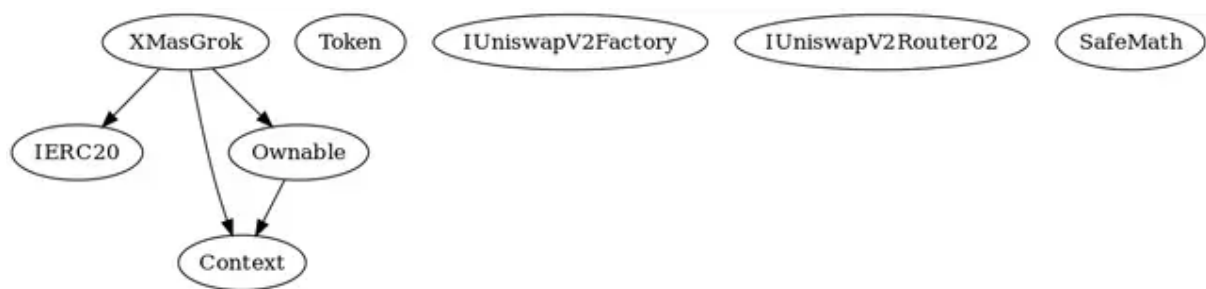
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Token	Interface			
	transferFrom	External	✓	-
	transfer	External	✓	-
IUniswapV2Factory	Interface			
	createPair	External	✓	-
IUniswapV2Router02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-

	WETH	External		-
	addLiquidityETH	External	Payable	-
Context	Implementation			
	_msgSender	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
XMasGrok	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-

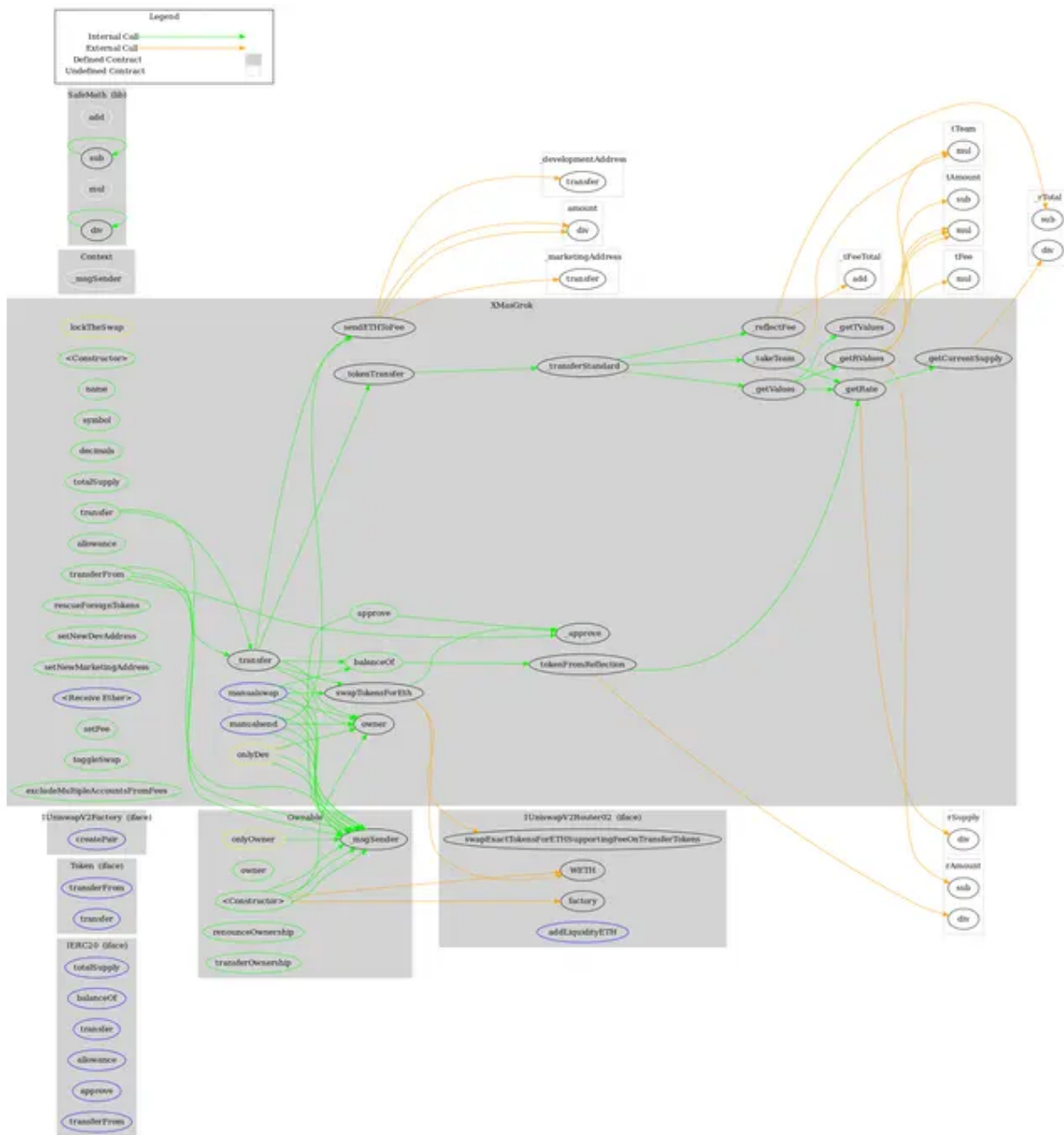
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	tokenFromReflection	Private		
	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	lockTheSwap
	sendETHToFee	Private	✓	
	_tokenTransfer	Private	✓	
	rescueForeignTokens	Public	✓	onlyDev
	setNewDevAddress	Public	✓	onlyDev
	setNewMarketingAddress	Public	✓	onlyDev
	_transferStandard	Private	✓	
	_takeTeam	Private	✓	
	_reflectFee	Private	✓	
		External	Payable	-
	_getValues	Private		
	_getTValues	Private		

	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	manualswap	External	✓	-
	manualsend	External	✓	-
	setFee	Public	✓	onlyDev
	toggleSwap	Public	✓	onlyDev
	excludeMultipleAccountsFromFees	Public	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Xmas Grok contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 16% fees.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>