Cyberscope

Audit Report

Grok X Ai

December 2023

Network	BSC
Address	0xf875af40467bd46bb78df8dc9bf805e04e6c11b3
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
●	RSD	Redundant Struct Declaration	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	GROKXAI
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://bscscan.com/address/0xf875af40467bd46bb78df8dc9bf805e04e6c11b3
Address	0xf875af40467bd46bb78df8dc9bf805e04e6c11b3
Network	BSC
Symbol	GROK X AI
Decimals	9
Total Supply	1,000,000,000

Audit Updates

Initial Audit	02 Dec 2023

Source Files

Filename	SHA256
GROKXAI.sol	c919a0323d4b28ce39be0f1c723d2618f839278d781eb4e3d8a57731799d555d

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	9

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	9	0	0	0

RSD - Redundant Struct Declaration

Criticality	Minor / Informative
Location	GROKXAI.sol#L150,155,163
Status	Unresolved

Description

The contract declares three separate structs, namely `BuyTaxes` , `SellTaxes` , and `TotFeesPaidStruct` , all with identical properties (`rfi` and `marketing`). This redundancy introduces unnecessary complexity to the codebase and may lead to potential maintenance challenges.

```
    struct BuyTaxes {
        uint256 rfi;
        uint256 marketing;
    }

    struct SellTaxes {
        uint256 rfi;
        uint256 marketing;
    }

    struct TotFeesPaidStruct {
        uint256 rfi;
        uint256 marketing;
    }
```

Recommendation

The team is advised to streamline and simplify the code by declaring a single struct that encompasses all properties shared by the three redundant structs (BuyTaxes, SellTaxes, and TotFeesPaidStruct). This consolidated struct can be used uniformly throughout the contract, eliminating redundancy and promoting code clarity. By consolidating the struct declarations, the contract can enhance readability, reduce the risk of inconsistencies, and facilitate future updates or modifications. The consolidated struct should contain all necessary properties used across different parts of the contract.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	GROKXAI.sol#L297
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
    function excludeFromReward(address account) private
onlyOwner {
        require(!_isExcluded[account], "Account is already
excluded");
        if (_rOwned[account] > 0) {
            _tOwned[account] =
tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    }


    function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	GROKXAI.sol#L307
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	GROKXAI.sol#L188
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
  constructor(address routerAddress) {
        IRouter _router = IRouter(routerAddress);
        address _pair =
IFactory(_router.factory()).createPair(address(this),
_router.WETH());

        router = _router;
        pair = _pair;
    ...
    }
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	GROKXAI.sol#L187
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.

```
pair
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	GROKXAI.sol#L135,138,140,141
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private _tTotal = 1 *10**9 * 10**_decimals
uint256 public swapTokensAtAmount = 20000 * 10**_decimals
address public deadWallet =
0x000000000000000000000000000000000000dEaD
address public marketingWallet =
0xE1BD10ba60D1CB89cC00cc51e67a547EdD0570C3
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	GROKXAI.sol#L80,132,143,144,166
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);
uint8 private constant _decimals = 9
string private constant _name = "Grok X Ai"
string private constant _symbol = "GROK X AI"

struct valuesFromGetValues {
        uint256 rAmount;
        uint256 rTransferAmount;
        uint256 rRfi;
        uint256 rMarketing;
        uint256 tTransferAmount;
        uint256 tRfi;
        uint256 tMarketing;
    }
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	GROKXAI.sol#L187
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.

```
pair = _pair
```

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	GROKXAI.sol#L5
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 ^0.8.17;
```

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.

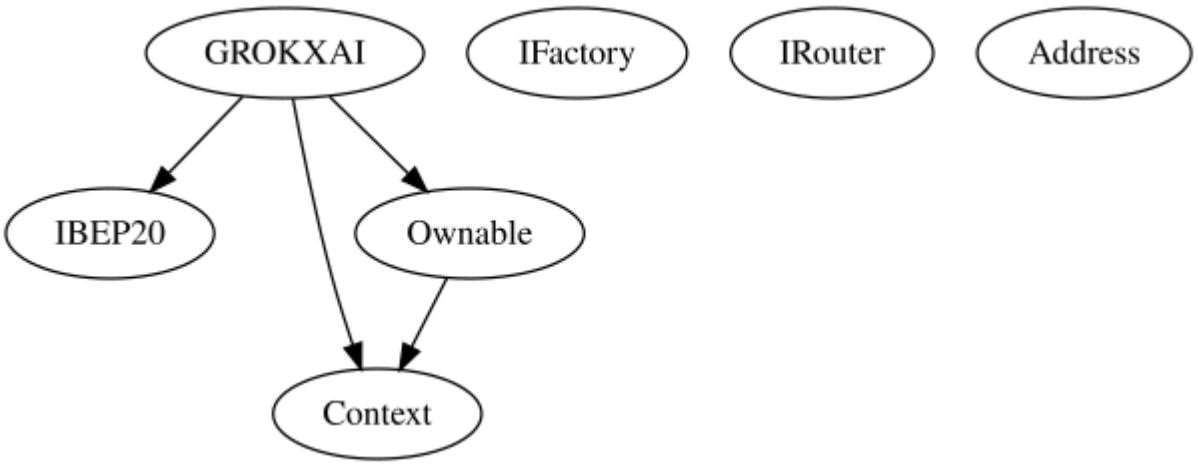
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	

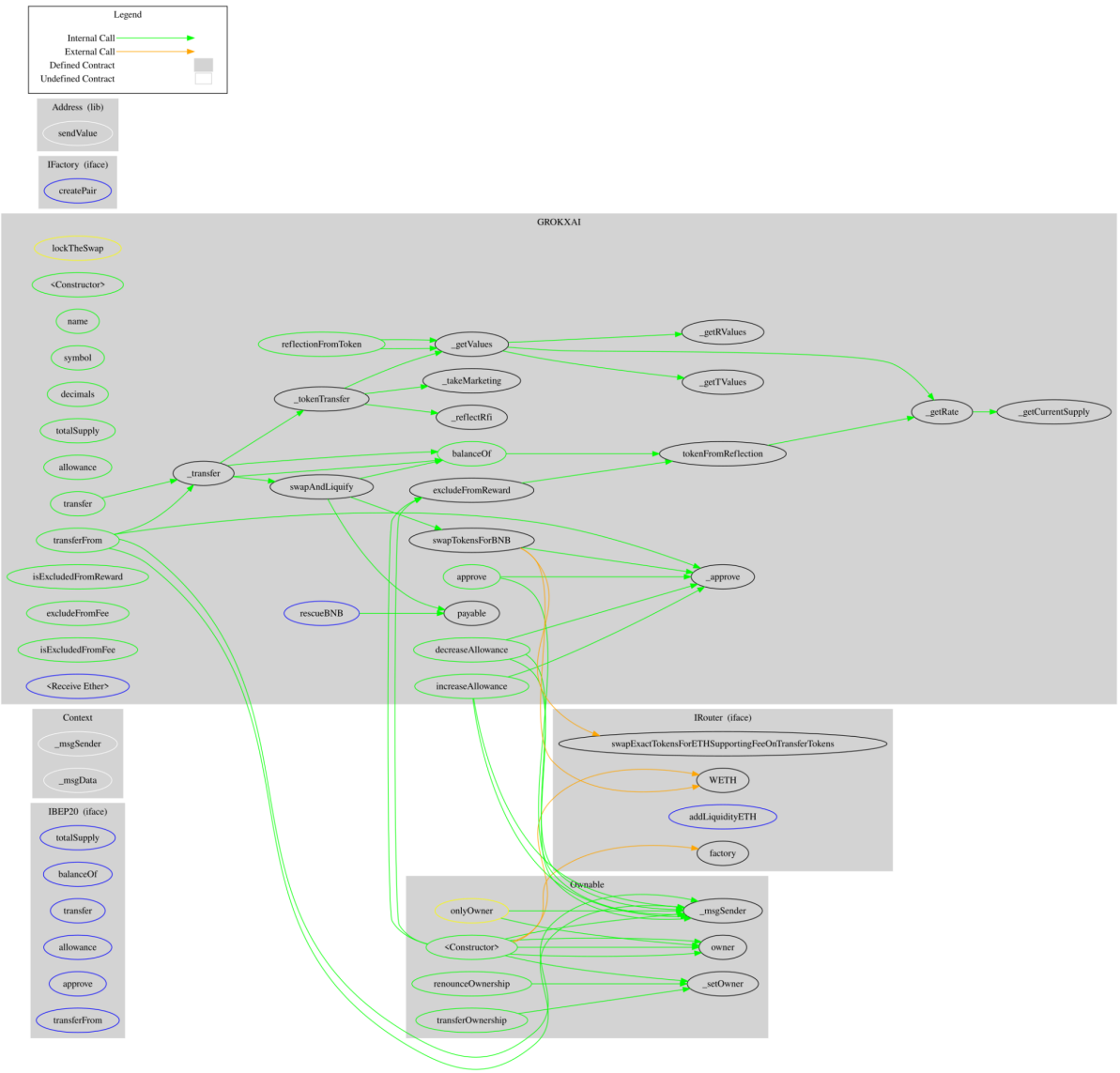
IFactory	Interface			
	createPair	External	✓	-
IRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
Address	Library			
	sendValue	Internal	✓	
GROKXAI	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-

	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	transfer	Public	✓	-
	isExcludedFromReward	Public		-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Private	✓	onlyOwner
	excludeFromFee	Public	✓	onlyOwner
	isExcludedFromFee	Public		-
	_reflectRfi	Private	✓	
	_takeMarketing	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_approve	Private	✓	
	_transfer	Private	✓	
	_tokenTransfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForBNB	Private	✓	
	rescueBNB	External	✓	onlyOwner
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Address (lib)
sendValue
IFactory (iface)
createPair
GROKXAI
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
allowance
transfer
transferFrom
isExcludedFromReward
excludeFromFee
isExcludedFromFee
<Receive Ether>
reflectionFromToken
_getValues
_getRValues
_getTValues
_takeMarketing
_tokenTransfer
_reflectRfi
_getRate
_getCurrentSupply
balanceOf
tokenFromReflection
_transfer
swapAndLiquify
excludeFromReward
swapTokensForBNB
approve
_approve
rescueBNB
payable
decreaseAllowance
increaseAllowance
Context
_msgSender
_msgData
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidityETH
factory
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
owner
_setOwner

Summary

Grok X Ai contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Grok X Ai is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. The fees are fixed at 5% for buys and sales and 10% for transfers.

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