Cyberscope

Audit Report

Legacy Of Game

March 2024

Network	BSC
Address	0xcf29be0798dc6c1eac6d427dcef554362ccbac23
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	Unresolved
	MT	Mints Tokens	Passed
	BT	Burns Tokens	Passed
	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	PBV	Percentage Boundaries Validation	Unresolved
●	RED	Redudant Event Declaration	Unresolved
●	RPK	Redundant Payable Keyword	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L22	Potential Locked Ether	Unresolved

Table of Contents

Review

Contract Name	LogX
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://bscscan.com/address/0xcf29be0798dc6c1eac6d427dcef554362ccbac23
Address	0xcf29be0798dc6c1eac6d427dcef554362ccbac23
Network	BSC
Symbol	LX
Decimals	18
Total Supply	8,800,000,000
Badge Eligibility	Must Fix Criticals

Audit Updates

Initial Audit	14 Mar 2024

Source Files

Filename	SHA256
LogX.sol	90b5eeebc2acd51ce2346ebed55ceaec1010dedc7b978ba2a31968089e88e506

Findings Breakdown

Critical	1
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Minor / Informative
Location	LogX.sol#L899,903
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users including the owner. The owner may stop the transactions by calling the `pause` function.

```
    function pause() external onlyOwner {
        _pause();
    }

    function unpause() external onlyOwner {
        _unpause();
    }
```

Recommendation

It is recommended to consider the removal of the pause functionality. Additionally, 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.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	LogX.sol#L933,943
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setBuyFee` and `setSellFee functions` with a high percentage value.

```
    function setBuyFee(uint256 value) external payable onlyOwner {
        _buyFee = value;
        emit BuyFeeSet(value);
    }

    function setSellFee(uint256 value) external payable onlyOwner {
        _sellFee = value;
        emit SellFeeSet(value);
    }
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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.

PBV - Percentage Boundaries Validation

Criticality	Minor / Informative
Location	LogX.sol#L952,957
Status	Unresolved

Description

The contract utilizes variables for percentage-based calculations that are required for its operations. These variables are involved in multiplication and division operations to determine proportions related to the contract's logic. If such variables are set to values beyond their logical or intended maximum limits, it could result in incorrect calculations. This misconfiguration has the potential to cause unintended behavior or financial discrepancies, affecting the contract's integrity and the accuracy of its calculations.

```
uint256 buyFees = (value * _buyFee) / THOUSANDTH_UNIT;
...
uint256 sellFees = (value * _sellFee) / THOUSANDTH_UNIT;
```

Recommendation

To mitigate risks associated with boundary violations, it is important to implement validation checks for variables used in percentage-based calculations. Ensure that these variables do not exceed their maximum logical values. This can be accomplished by incorporating `require` statements or similar validation mechanisms whenever such variables are assigned or modified. These safeguards will enforce correct operational boundaries, preserving the contract's intended functionality and preventing computational errors.

RED - Redudant Event Declaration

Criticality	Minor / Informative
Location	LogX.sol#L828
Status	Unresolved

Description

The contract uses events that are not emitted within the contract's functions. As a result, these declared events are redundant and serve no purpose within the contract's current implementation.

```
event TreasurySet(address treasury);
 event IncludedInFee(address included);
 event ExcludedFromFee(address excluded);
```

Recommendation

To optimize contract performance and efficiency, it is advisable to regularly review and refactor the codebase, removing the unused event declarations. This proactive approach not only streamlines the contract, reducing deployment and execution costs but also enhances readability and maintainability.

RPK - Redundant Payable Keyword

Criticality	Minor / Informative
Location	LogX.sol#L912,922,932,942
Status	Unresolved

Description

The contract is utilizing the `payable` keyword in several functions that are exclusively callable by the contract owner, such as `setPair`, `unsetPair`, `setBuyFee`, and `setSellFee`. These functions are designed to modify contract state variables or settings and do not involve transferring ether to the contract. Given that these functions can only be executed by the owner and do not inherently require the transfer of ether to be completed successfully, the inclusion of the payable keyword is unnecessary and redundant. This misalignment does not directly impact the security or functionality of the contract but does introduce potential confusion regarding the intended use and capabilities of these functions. It suggests that ether could be sent along with the transaction, which is not a requirement or utilized aspect of these functions.

```
    function setPair(address pair) external payable onlyOwner {
        _pairs[pair] = true;
        emit PairSet(pair);
    }

    function unsetPair(address pair) external payable onlyOwner {
        ...
    }

    function setBuyFee(uint256 value) external payable onlyOwner {
        ...
    }

    function setSellFee(uint256 value) external payable onlyOwner {
        ...
    }
```

Recommendation

It is recommended to remove the `payable` keyword from the declarations of `setPair`, `unsetPair`, `setBuyFee`, and `setSellFee` functions. This change would clarify the intended use of these functions, ensuring that their design aligns with their actual capabilities and requirements. Removing the `payable` keyword where it is not needed can also prevent confusion and potential misuse by indicating that these functions are not intended to receive ether. Additionally, this adjustment would contribute to cleaner, more understandable code, adhering to best practices for smart contract development.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	LogX.sol#L912,922
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 setPair(address pair) external payable onlyOwner {
        _pairs[pair] = true;
        emit PairSet(pair);
    }

    function unsetPair(address pair) external payable onlyOwner
{
        _pairs[pair] = false;
        emit PairUnset(pair);
    }
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	LogX.sol#L277,625
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
    function _contextSuffixLength() internal view virtual
returns (uint256) {
        return 0;
    }

    function _burn(address account, uint256 value) internal {
        if (account == address(0)) {
            revert ERC20InvalidSender(address(0));
        }
        _update(account, address(0), value);
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	LogX.sol#L912,922,932,942
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
function setPair(address pair) external payable onlyOwner
function unsetPair(address pair) external payable onlyOwner
function setBuyFee(uint256 value) external payable onlyOwner
function setSellFee(uint256 value) external payable onlyOwner
```

Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

Functions Analysis

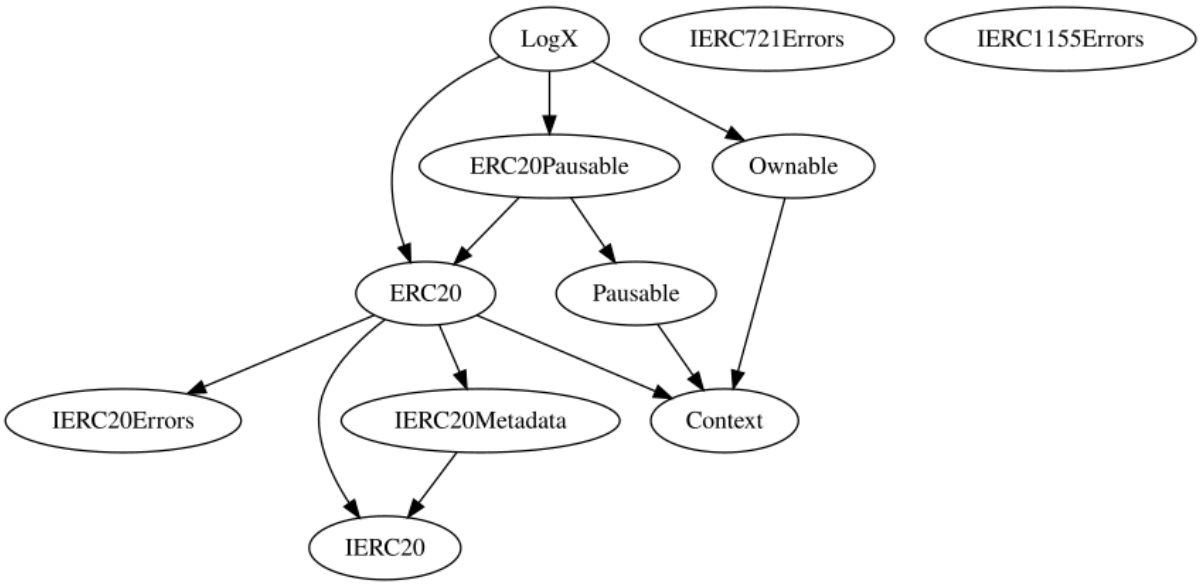
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-

Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
ERC20	Implementation	Context, IERC20, IERC20Metadata, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-

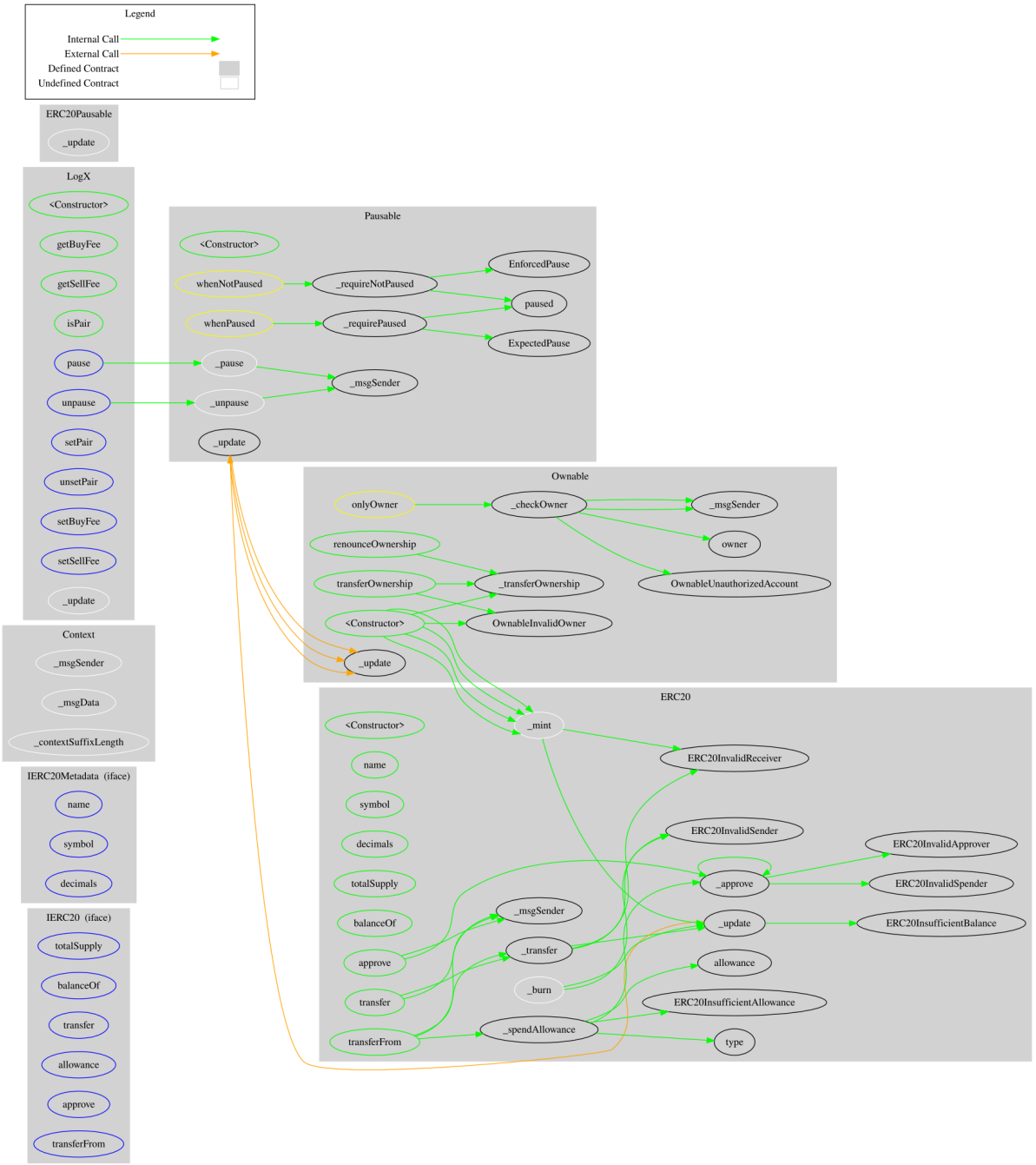
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
ERC20Pausable	Implementation	ERC20, Pausable		
	_update	Internal	✓	whenNotPaused
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

LogX	Implementation	ERC20, ERC20Pausable, Ownable		
		Public	✓	ERC20 Ownable
	getBuyFee	Public		-
	getSellFee	Public		-
	isPair	Public		-
	pause	External	✓	onlyOwner
	unpause	External	✓	onlyOwner
	setPair	External	Payable	onlyOwner
	unsetPair	External	Payable	onlyOwner
	setBuyFee	External	Payable	onlyOwner
	setSellFee	External	Payable	onlyOwner
	_update	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20Pausable
_update
LogX
<Constructor>
getBuyFee
getSellFee
isPair
pause
unpause
setPair
unsetPair
setBuyFee
setSellFee
_update
Context
_msgSender
_msgData
_contextSuffixLength
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Pausable
<Constructor>
whenNotPaused
_requireNotPaused
EnforcedPause
paused
whenPaused
_requirePaused
ExpectedPause
_pause
_msgSender
_unpause
_update
Ownable
onlyOwner
_checkOwner
_msgSender
owner
renounceOwnership
transferOwnership
_transferOwnership
OwnableUnauthorizedAccount
<Constructor>
OwnableInvalidOwner
_update
ERC20
<Constructor>
_mint
ERC20InvalidReceiver
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_msgSender
_transfer
_burn
_spendAllowance
ERC20InvalidSender
_approve
ERC20InvalidApprover
ERC20InvalidSpender
_update
ERC20InsufficientBalance
allowance
ERC20InsufficientAllowance
type

Summary

Legacy Of Game 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 and manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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