Cyberscope

Audit Report

XRP589

August 2023

Network ETH

Address 0xb63F7284f65eecD25Dad09Bf04AD2b849fC8AE42

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
	RVU	Redundant Variable Usage	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	MEE	Missing Events Emission	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	Unresolved
	L13	Divide before Multiply Operation	Unresolved
	L15	Local Scope Variable Shadowing	Unresolved

Table of Contents

Review

Contract Name	XRP589
Compiler Version	v0.8.20+commit.a1b79de6
Optimization	200 runs
Explorer	https://etherscan.io/address/0xb63f7284f65eecd25dad09bf04ad2b849fc8ae42
Address	0xb63f7284f65eecd25dad09bf04ad2b849fc8ae42
Network	ETH
Symbol	XRP589
Decimals	18
Total Supply	58,900

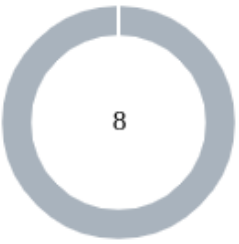
Audit Updates

Initial Audit	24 Aug 2023

Source Files

Filename	SHA256
XRP589.sol	742344330920017f29371e8c1f42f04197fb3564197fc01b04e571a2fc37d9b0

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	8	0	0	0

RVU - Redundant Variable Usage

Criticality	Minor / Informative
Location	XRP589.sol#L296
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract contains the variables `buyMarketingFee` , `buyTotalFees` , `sellMarketingFee` , and `sellTotalFees` . However, the variables `buyTotalFees` and `sellTotalFees` are redundant. Specifically, `buyTotalFees` is always set to be equal to `buyMarketingFee` , and similarly, `sellTotalFees` is always set to be equal to `sellMarketingFee` . As a result, these variables do not serve any distinct purpose and merely duplicate the values of `buyMarketingFee` and `sellMarketingFee` respectively. This redundancy could lead to unnecessary gas consumption and could make the contract more complex than it needs to be.

```
buyMarketingFee = 3;
buyTotalFees = buyMarketingFee;
...
sellMarketingFee = 3;
sellTotalFees = sellMarketingFee;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommended to remove the redundant variables `buyTotalFees` and `sellTotalFees` from the contract. Instead, directly use the corresponding variables `buyMarketingFee` and `sellMarketingFee` wherever the total fees for buying and selling are required.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	XRP589.sol#L378
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract modifies the state of certain variables without checking if their current state is equal to the one given as an argument. As a result, the contract performs redundant storage writes.

```
    function toggleSwap(bool value) external onlyOwner {
         swapEnabled = value;
     }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	XRP589.sol#L371,381
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 updateBuyFees(uint256 _marketingFee) external
onlyOwner {
        buyMarketingFee = _marketingFee;
        buyTotalFees = buyMarketingFee;
        require(buyTotalFees <= 5, "Must keep fees at 5% or
less");
    }

    function updateSellFees(uint256 _marketingFee) external
onlyOwner {
        sellMarketingFee = _marketingFee;
        sellTotalFees = sellMarketingFee;
        require(sellTotalFees <= 5, "Must keep fees at 5% or
less");
    }
```

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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	XRP589.sol#L282,285
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.

```
dexRouter
lpPair
```

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	XRP589.sol#L191,259,371,381,496,510
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);
mapping (address => bool) public
_isExcludedMaxTransactionAmount
uint256 _marketingFee
address _token
address _to
address _marketingAddress

function SetMarketingWallet(address _marketingAddress) external
onlyOwner {
        require(_marketingAddress != address(0),
"_marketingAddress address cannot be 0");
        marketingAddress = payable(_marketingAddress);
    }
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	XRP589.sol#L343,372,382
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
swapTokensAtAmount = newAmount
buyMarketingFee = _marketingFee
sellMarketingFee = _marketingFee
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	XRP589.sol#L438,443,444
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
fees = amount * buyTotalFees / 100
tokensForMarketing += fees * buyMarketingFee / buyTotalFees
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	XRP589.sol#L289
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
uint256 totalSupply = 58900  * (10**18)
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

Functions Analysis

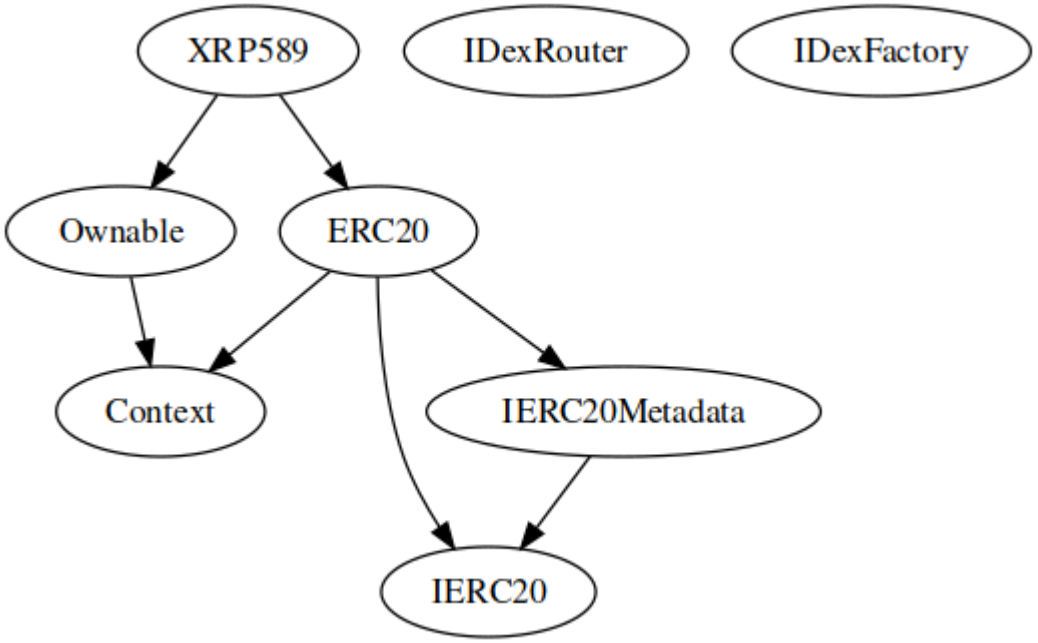
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-

ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

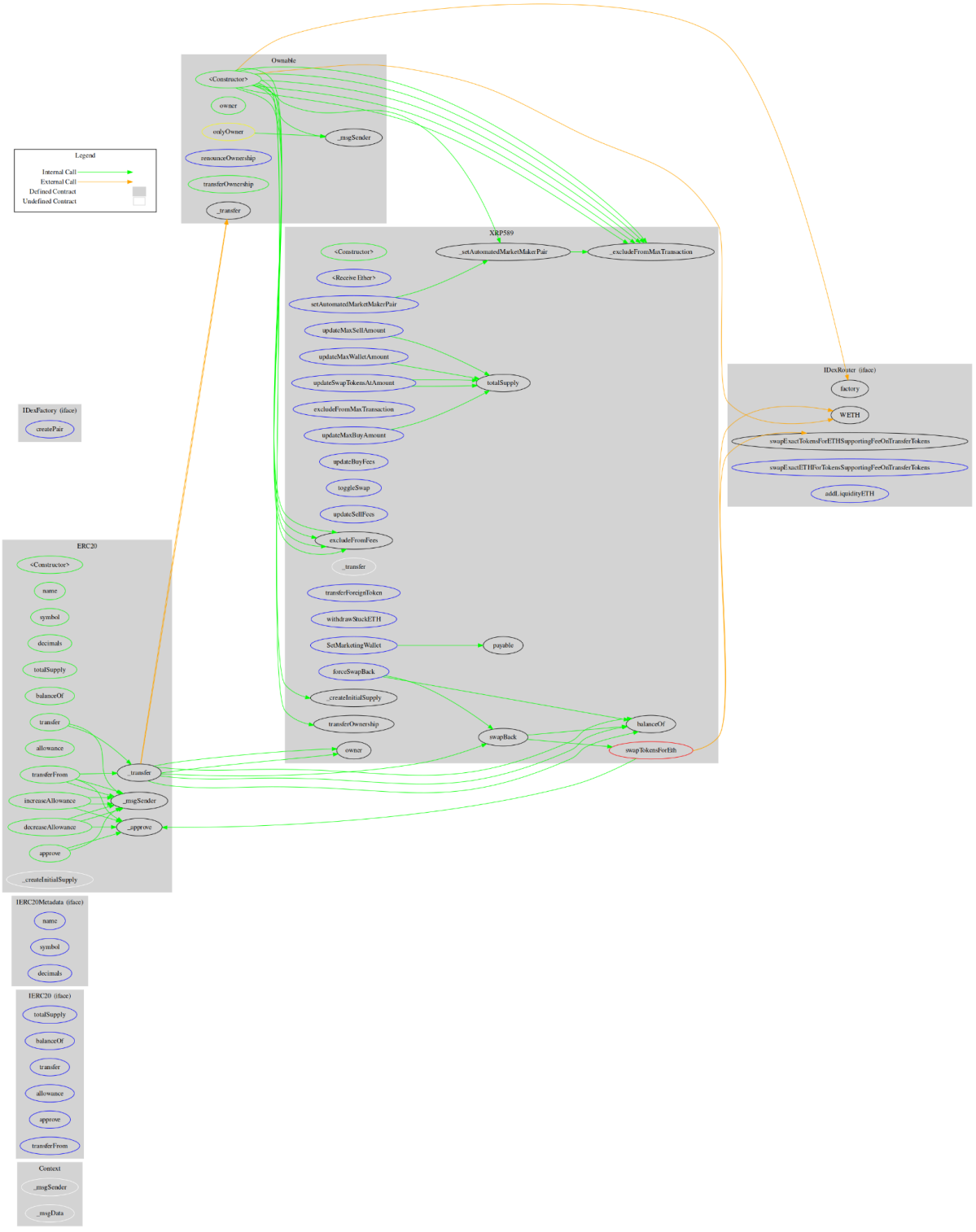
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	addLiquidityETH	External	Payable	-
IDexFactory	Interface			
	createPair	External	✓	-
XRP589	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	updateMaxBuyAmount	External	✓	onlyOwner
	updateMaxSellAmount	External	✓	onlyOwner
	updateMaxWalletAmount	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner
	_excludeFromMaxTransaction	Private	✓	
	excludeFromMaxTransaction	External	✓	onlyOwner
	setAutomatedMarketMakerPair	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateBuyFees	External	✓	onlyOwner

	toggleSwap	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	swapBack	Private	✓	
	transferForeignToken	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	SetMarketingWallet	External	✓	onlyOwner
	forceSwapBack	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
_transfer
_msgSender
Legend
Internal Call
External Call
Defined Contract
Undefined Contract
XRP589
<Constructor>
<Receive Ether>
setAutomatedMarketMakerPair
updateMaxSellAmount
updateMaxWalletAmount
updateSwapTokensAtAmount
excludeFromMaxTransaction
updateMaxBuyAmount
updateBuyFees
toggleSwap
updateSellFees
excludeFromFees
_transfer
transferForeignToken
withdrawStuckETH
SetMarketingWallet
forceSwapBack
_createInitialSupply
transferOwnership
owner
_setAutomatedMarketMakerPair
excludeFromMaxTransaction
totalSupply
payable
swapBack
balanceOf
swapTokensForEth
IDexRouter (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
addLiquidityETH
IDexFactory (iface)
createPair
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_createInitialSupply
_transfer
_msgSender
_approve
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

Summary

R Dev contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. R Dev 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. There is also a limit of max 5% 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