Cyberscope

Audit Report

Bitcointry Token

January 2024

Network	BSC
Address	0x2024b9be6b03f2a57d3533ae33c7e1d0b0b4be47
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
	CO	Code Optimization	Unresolved
	RFS	Redundant Fee Structure	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L13	Divide before Multiply Operation	Unresolved
	L22	Potential Locked Ether	Unresolved

Table of Contents

Review

Contract Name	Bitcointry_Token
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x2024b9be6b03f2a57d3533ae33c7e1d0b0b4be47
Address	0x2024b9be6b03f2a57d3533ae33c7e1d0b0b4be47
Network	BSC
Symbol	BTTY
Decimals	18
Total Supply	497,420,000
Badge Eligibility	Yes

Audit Updates

Initial Audit	13 Jan 2024

Source Files

Filename	SHA256
TokenRecover.sol	8e9398635a7efa71f68b9bc847e9476fba96c310cba12235fc3060af11e28571
Token.sol	da285962076df08746401472d9a9f22305fa99e1a91eb88f22ef627d896f2597

Ownable.sol	33422e7771fefe5fbfe8934837515097119d82a50eda0e49b38e4d6a64a1c25d
Initializable.sol	b05c26d897c4178cbdb35ad113527e463e1bdeae5764869318a54f93c8b98a94
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC20Metadata.sol	b10e2f8bcc3ed53a5d9a82a29b1ad3209225331bb4de4a0459862a762cf83a1a
IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
ERC20Burnable.sol	480b22ce348050fdb85a693e38ed6b4767a94e4776fc6806d6808a0ec171177e
ERC20.sol	f70c6ae5f2dda91a37e17cfcbec390cc59515ed0d34e316f036f5431b5c0a3f2
Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

CO - Code Optimization

Criticality	Minor / Informative
Location	Token.sol#L106,161,167,174
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.

Specifically, `_beforeTokenTransfer` and `_afterTokenTransfer` override functions merely call their respective super implementations without adding any additional functionality. Overriding these functions in the current state does not contribute any value, as they do not extend or modify the base behavior.

Additionally, the function `_setAMMPair` contains an empty conditional statement. This empty block suggests an incomplete implementation or an unnecessary code segment, resulting in potential confusion regarding its purpose.

Lastly, in the `_transfer` function, the boolean _swapping is initialized but never set to true within the contract. Its presence in the conditional check if `(!_swapping && ...)` is therefore redundant, as it does not actively contribute to the function's logic.

```
function _beforeTokenTransfer(address from, address to, uint256
amount)
    internal
    override
{
    super._beforeTokenTransfer(from, to, amount);
}

function _afterTokenTransfer(address from, address to, uint256
amount)
    internal
    override
{
    super._afterTokenTransfer(from, to, amount);
}

function _setAMMPair(address pair, bool isPair) private {
    AMMPairs[pair] = isPair;

    if (isPair) {
    }

    emit AMMPairsUpdated(pair, isPair);
}

bool private _swapping;

function _transfer(
...
    (!_swapping && ...)
    ...
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them or remove 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.

RFS - Redundant Fee Structure

Criticality	Minor / Informative
Location	Token.sol#L26,30,84
Status	Unresolved

Description

Both `totalFees` and `stakingFees` arrays are declared and utilized in a manner that results in them always holding identical values. This redundancy is evident in the `stakingFeesSetup` function, where `totalFees` are recalculated based on the current values of `stakingFees`. Initially, as both arrays are initialized to zeros, the first call to stakingFeesSetup sets `totalFees` to the same values as `stakingFees`. Subsequent calls maintain this equality, as `totalFees` is reset to mirror the latest `stakingFees`. The current implementation does not reflect an accumulation or a diverse set of fees; instead, it merely duplicates the values from `stakingFees`. This redundancy raises questions about the necessity of maintaining two separate arrays for fees or potential issues with the business logic, as they do not provide distinct data, contrary to what might be expected from their naming and structure.

```
address public stakingAddress;

uint16[3] public totalFees;

function stakingFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
    totalFees[0] = totalFees[0] - stakingFees[0] + _buyFee;
    totalFees[1] = totalFees[1] - stakingFees[1] + _sellFee;
    totalFees[2] = totalFees[2] - stakingFees[2] +
_transferFee;
    require(totalFees[0] <= 2500 && totalFees[1] <= 2500 &&
totalFees[2] <= 2500, "TaxesDefaultRouter: Cannot exceed max
total fee of 25%");

    stakingFees = [_buyFee, _sellFee, _transferFee];

    emit stakingFeesUpdated(_buyFee, _sellFee, _transferFee);
}
```

Recommendation

It is recommended to reevaluate the purpose and usage of the `totalFees` and `stakingFees` arrays. The current mechanism results in both arrays always holding identical values, leading to an unnecessary duplication of data. If the intent was for totalFees to represent a cumulative or diverse set of fees separate from `stakingFees`, the logic in `stakingFeesSetup` needs to be revised to reflect this.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L23,35,37,38,39,61,75,84
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.

```
contract Bitcointry_Token is ERC20, ERC20Burnable, Ownable,
TokenRecover, Initializable {

    address public stakingAddress;
    uint16[3] public stakingFees;

    mapping (address => bool) public isExcludedFromFees;
...
        internal
        override
    {
        super._afterTokenTransfer(from, to, amount);
    }
}


...
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Token.sol#L122,126
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 * totalFees[txType] / 10000;

stakingPortion = fees * stakingFees[txType] /
totalFees[txType];
```

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.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	Token.sol#L65
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.

```
receive() external payable {}
```

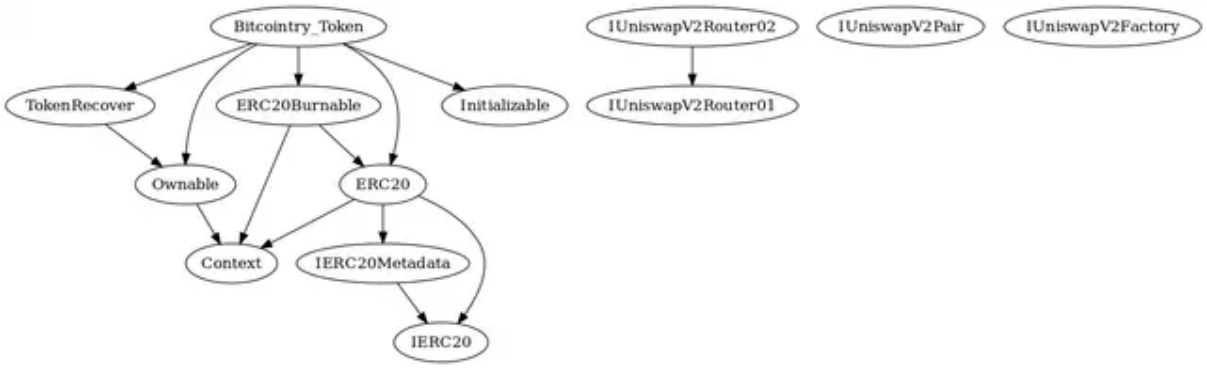
Recommendation

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.

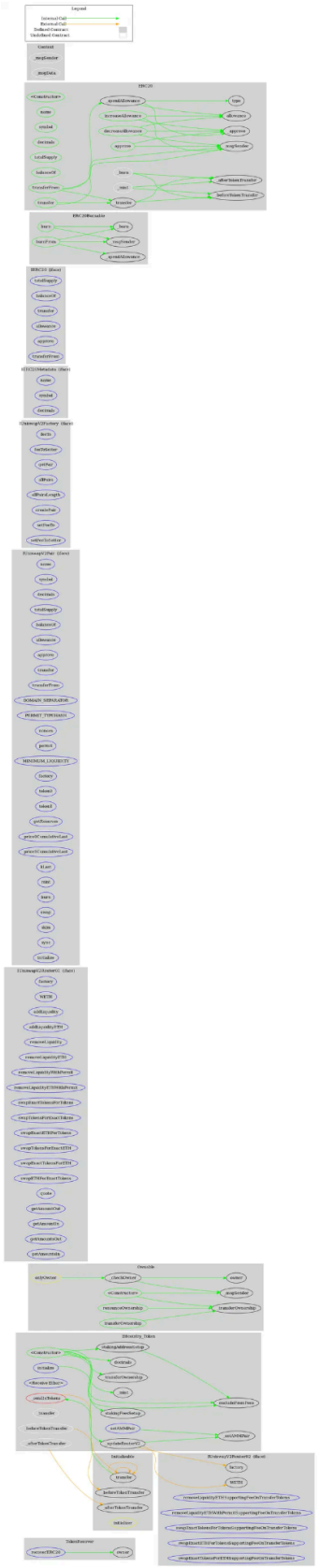
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Bitcointry_Token	Implementation	ERC20, ERC20Burnable, Ownable, TokenRecover, Initializable		
		Public	✓	ERC20
	initialize	External	✓	initializer
		External	Payable	-
	decimals	Public		-
	_sendInTokens	Private	✓	
	stakingAddressSetup	Public	✓	onlyOwner
	stakingFeesSetup	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_updateRouterV2	Private	✓	
	setAMMPair	External	✓	onlyOwner
	_setAMMPair	Private	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	

Inheritance Graph



Flow Graph



Summary

Bitcointry Token contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Bitcointry Token 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 25% 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