



Cyberscope

# Audit Report

## **Epstein Coin**

May 2023

GitHub      <https://github.com/Kinco-dev/Epstein-Coin-JEFFY>  
Commit      9e6c1b62eb16a8d05818e555d4190fdd3350d68e  
Audited by   © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Review</b>	<b>2</b>
Audit Updates	2
Source Files	2
<b>Findings Breakdown</b>	<b>3</b>
<b>Analysis</b>	<b>4</b>
OCTD - Transfers Contract's Tokens	5
Description	5
Recommendation	5
<b>Diagnostics</b>	<b>6</b>
TUU - Time Units Usage	7
Description	7
Recommendation	7
RM - Redundant Multiplication	8
Description	8
Recommendation	8
L04 - Conformance to Solidity Naming Conventions	9
Description	9
Recommendation	9
L08 - Tautology or Contradiction	10
Description	10
Recommendation	10
L09 - Dead Code Elimination	11
Description	11
Recommendation	12
<b>Functions Analysis</b>	<b>13</b>
<b>Inheritance Graph</b>	<b>18</b>
<b>Flow Graph</b>	<b>19</b>
<b>Summary</b>	<b>20</b>
<b>Disclaimer</b>	<b>21</b>
<b>About Cyberscope</b>	<b>22</b>

## Review

Repository	<a href="https://github.com/Kinco-dev/Epstein-Coin-JEFFY">https://github.com/Kinco-dev/Epstein-Coin-JEFFY</a>
Commit	9e6c1b62eb16a8d05818e555d4190fdd3350d68e
Testing Deploy	<a href="https://testnet.bscscan.com/address/0x72cf071F97d8fd79cA2AdbB247939Fb51E18452">https://testnet.bscscan.com/address/0x72cf071F97d8fd79cA2AdbB247939Fb51E18452</a>
Address	0x72cf071F97d8fd79cA2AdbB247939Fb51E18452c

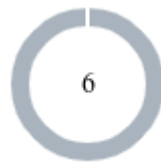
## Audit Updates

Initial Audit	17 May 2023
---------------	-------------

## Source Files

Filename	SHA256
epstein_coin.sol	91b468ab8a31a920f6f3d5f8b927fd7be4bad3f5715716d256552514248e2601

## Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	6	0	0	0

# Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

## OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	epstein_coin.sol#L511
Status	Unresolved

### Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `withdrawStuckERC20Tokens` function.

```
function withdrawStuckERC20Tokens(address token, address to) external
onlyOwner {
    require(IERC20(token).balanceOf(address(this)) > 0, "JEFRY: There are
no tokens in the contract");
    require(IERC20(token).transfer(to,
IERC20(token).balanceOf(address(this))));
}
```

### 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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

## Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TUU	Time Units Usage	Unresolved
●	RM	Redundant Multiplication	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L09	Dead Code Elimination	Unresolved

## TUU - Time Units Usage

Criticality	Minor / Informative
Location	epstein_coin.sol#L430
Status	Unresolved

### Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
require(timeInSeconds <= 600, "JEFRY: The cooldown must be lower or equals  
to 600 seconds");
```

### Recommendation

It is a good practice to use the time units reserved keywords like `seconds`, `minutes`, `hours`, `days`, `weeks` and `years` to process time-related calculations.

It's important to note that these time units are simply a shorthand notation for representing time in seconds, and do not have any effect on the actual passage of time or the execution of the contract. The time units are simply a convenience for expressing time in a more human-readable form.



## RM - Redundant Multiplication

<b>Criticality</b>	Minor / Informative
<b>Location</b>	epstein_coin.sol#L431
<b>Status</b>	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 multiplies the variable `timeInSeconds` by 1. This multiplication will not modify the result. As a result, the multiplication is redundant.

```
coolDownTime = timeInSeconds * 1 seconds;
```

### 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.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	epstein_coin.sol#L258,287,496
<b>Status</b>	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);
address[] calldata _holders
uint256[] calldata _amounts
```

### Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and 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>.

## L08 - Tautology or Contradiction

<b>Criticality</b>	Minor / Informative
<b>Location</b>	epstein_coin.sol#L461
<b>Status</b>	Unresolved

### Description

A tautology is a logical statement that is always true, regardless of the values of its variables. A contradiction is a logical statement that is always false, regardless of the values of its variables.

Using tautologies or contradictions can lead to unintended behavior and can make the code harder to understand and maintain. It is generally considered good practice to avoid tautologies and contradictions in the code.

```
require(amount >= 0, "JEFRY: Transfer amount must be greater or equals to zero")
```

### Recommendation

The team is advised to carefully consider the logical conditions is using in the code and ensure that it is well-defined and make sense in the context of the smart contract.

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	epstein_coin.sol#L202
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 _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    ...
    _totalSupply -= amount;
}

emit Transfer(account, address(0), amount);

_afterTokenTransfer(account, address(0), amount);
}
```

## 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.

## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>Address</b>	Library			
	sendValue	Internal	✓	
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>IERC20Metadata</b>	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
<b>Context</b>	Implementation			
	_msgSender	Internal		

	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
<b>ERC20</b>	Implementation	Context, IERC20, IERC20Meta data		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-

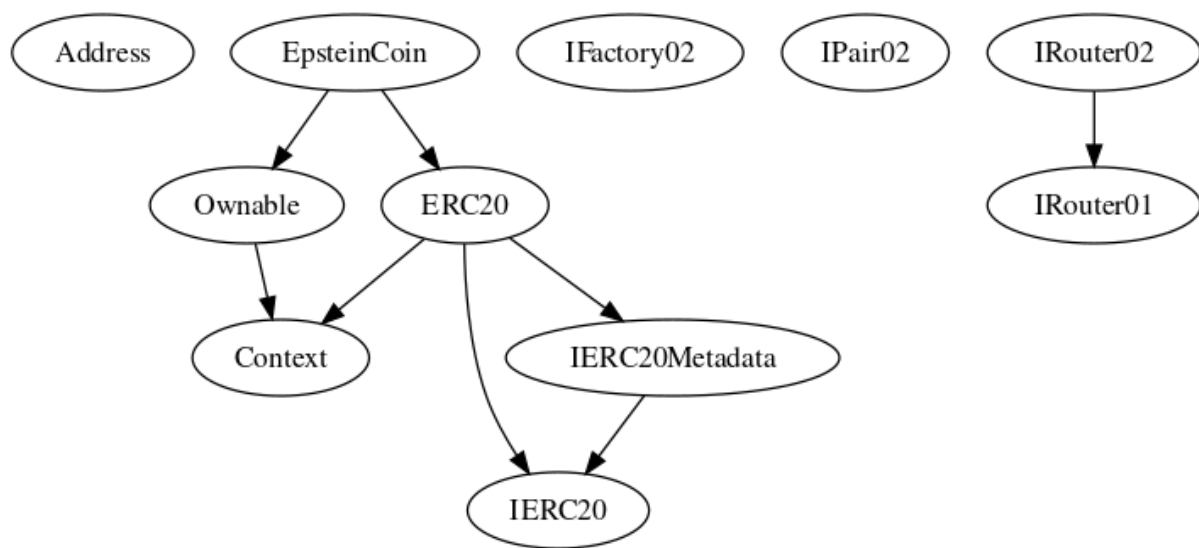
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
<b>IFactory02</b>	Interface			
	getPair	External		-
	createPair	External	✓	-
<b>IPair02</b>	Interface			
	factory	External		-
	getReserves	External		-
	sync	External	✓	-
<b>IRouter01</b>	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-



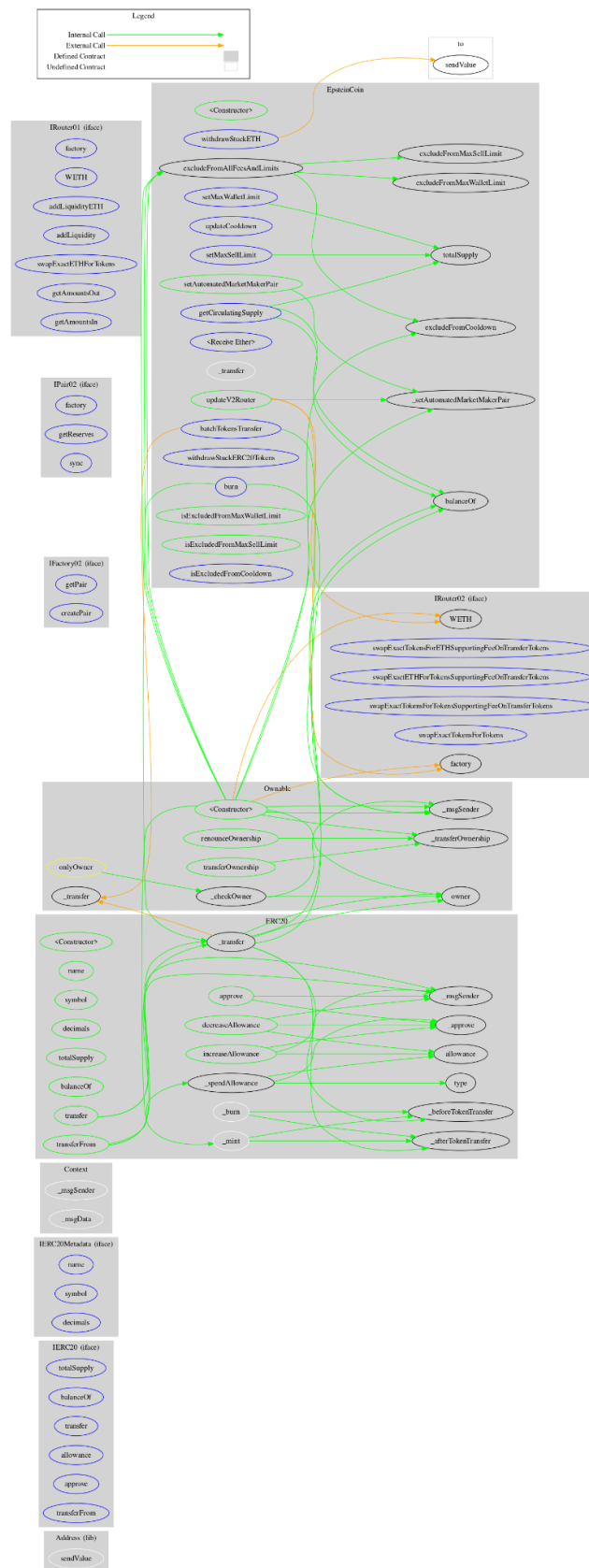
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
<b>IRouter02</b>	Interface	IRouter01		
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
<b>EpsteinCoin</b>	Implementation	ERC20, Ownable		
		Public	✓	ERC20
	excludeFromAllFeesAndLimits	Public	✓	onlyOwner
	excludeFromMaxWalletLimit	Public	✓	onlyOwner
	excludeFromMaxSellLimit	Public	✓	onlyOwner
	excludeFromCooldown	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateV2Router	Public	✓	onlyOwner
	updateCooldown	External	✓	onlyOwner
	setMaxWalletLimit	External	✓	onlyOwner

	setMaxSellLimit	External	✓	onlyOwner
	burn	External	✓	-
		External	Payable	-
	_transfer	Internal	✓	
	batchTokensTransfer	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	withdrawStuckERC20Tokens	External	✓	onlyOwner
	getCirculatingSupply	External		-
	isExcludedFromMaxWalletLimit	Public		-
	isExcludedFromMaxSellLimit	Public		-
	isExcludedFromCooldown	External		-

## Inheritance Graph



# Flow Graph



## Summary

Epstein Coin 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 drain the contract's tokens. 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>