Cyberscope

Audit Report

TrumpJr

March 2024

Network	ETH
Address	0xe137407dbf9a768f342de7b88607e186fc9c09fb
Audited by	© cyberscope

Table of Contents

Review

Contract Name	TaxableToken
Compiler Version	v0.8.20+commit.a1b79de6
Optimization	200 runs
Explorer	https://etherscan.io/address/0xe137407dbf9a768f342de7b88607e186fc9c09fb
Address	0xe137407dbf9a768f342de7b88607e186fc9c09fb
Network	ETH

Audit Updates

Initial Audit	05 Mar 2024

Source Files

Filename	SHA256
contracts/TaxableToken.sol	7f3fa68a0373b6c6fa4838ddf159af9a4bb80d2c89687a955caa26f71725a8b9
contracts/core/ERC20Taxable.sol	f063a00d4cb8a89429fe4d1537584807f4c5f1af7d60131b831422dca2613414
contracts/core/ERC20.sol	6c11180c5ee7f4c34dec33783f8e225cf6d787e4697cf359b623c429b931619f
contracts/core/utils/BlackList.sol	0fd6a5793c1302245c8a08123b8c56319d44fde95cdab6626c141a085c7d29ef
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

@openzeppelin/contracts/utils/Address.sol	8b85a2463eda119c2f42c34fa3d942b61aee65df381f48ed436fe8edb3a7d602
@openzeppelin/contracts/token/ERC20/IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/security/Pausable.sol	2072248d2f79e661c149fd6a6593a8a3f038466557c9b75e50e0b001bcb5cf97
@openzeppelin/contracts/proxy/utils/Initializable.sol	6166f84ded782b5c1f471fc4aed9ee3d1fb74f55ec79e920f50510392e235eaa
@openzeppelin/contracts/access/Ownable.sol	a8e4e1ae19d9bd3e8b0a6d46577eec098c01fbaffd3ec1252fd20d799e73393b

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	CR	Code Repetition	Unresolved
	MEE	Missing Events Emission	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L15	Local Scope Variable Shadowing	Unresolved
	L19	Stable Compiler Version	Unresolved

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/core/ERC20Taxable.sol#L45,76
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
   function transfer(address to, uint256 amount) public virtual override
returns (bool) {
        ...

        if (_taxRate > 0 && !(_isExcludedFromTaxFee[owner] ||
_isExcludedFromTaxFee[to])) {
            uint256 taxAmount = (amount * _taxRate) / 1000;

            if (taxAmount > 0) {
                _transfer(owner, _taxAddress, taxAmount);
                unchecked {
                    amount -= taxAmount;
                }
            }
        }

        _transfer(owner, to, amount);

        return true;
    }

    function transferFrom(address from, address to, uint256 amount)
public virtual override returns (bool) {
        ...

        if (_taxRate > 0 && !(_isExcludedFromTaxFee[from] ||
_isExcludedFromTaxFee[to])) {
            uint256 taxAmount = (amount * _taxRate) / 1000;

            if (taxAmount > 0) {
                _transfer(from, _taxAddress, taxAmount);
                unchecked {
                    amount -= taxAmount;
                }
            }
        }

        _transfer(from, to, amount);

        return true;
    }
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/TaxableToken.sol#L56,64 contracts/core/ERC20Taxable.sol#L122,149
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 setTaxRate(uint256 _newTaxFee) public onlyOwner {
        _setTaxRate(_newTaxFee);
    }

    function setExclusionFromTaxFee(address _account, bool
_status) public onlyOwner {
        _setExclusionFromTaxFee(_account, _status);
    }

    function _setTaxRate(uint256 taxFeePerMille_) internal
virtual {
        require(taxFeePerMille_ < 1000, "ERC20Taxable:
taxFeePerMille_ must be less than 1000");

        _taxRate = taxFeePerMille_;
    }

    function _setExclusionFromTaxFee(address account_, bool
status_) internal virtual {
        _isExcludedFromTaxFee[account_] = status_;
    }
```

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	contracts/TaxableToken.sol#L64 contracts/core/ERC20Taxable.sol#L149
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 setExclusionFromTaxFee(address _account, bool
_status) public onlyOwner {
        _setExclusionFromTaxFee(_account, _status);
    }

    function _setExclusionFromTaxFee(address account_, bool
status_) internal virtual {
        _isExcludedFromTaxFee[account_] = status_;
    }
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/TaxableToken.sol#L16,17,18,19,20,21,22,23,48,52,56,60,64 contracts/core/utils/BlackList.sol#L17
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.

```
address _owner
string memory _name
string memory _symbol
uint8 _decimals
uint256 _initialSupply
uint256 _maxSupply
uint256 _taxFeePerMille
address _taxAddress
address _account
uint256 _newTaxFee
address _newTaxAddress
bool _status
```

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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	contracts/TaxableToken.sol#L23
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.

```
address _taxAddress
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/TaxableToken.sol#L2 contracts/core/utils/BlackList.sol#L2 contracts/core/ERC20Taxable.sol#L2
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 ^0.8.19;
pragma solidity ^0.8.0;
```

Recommendation

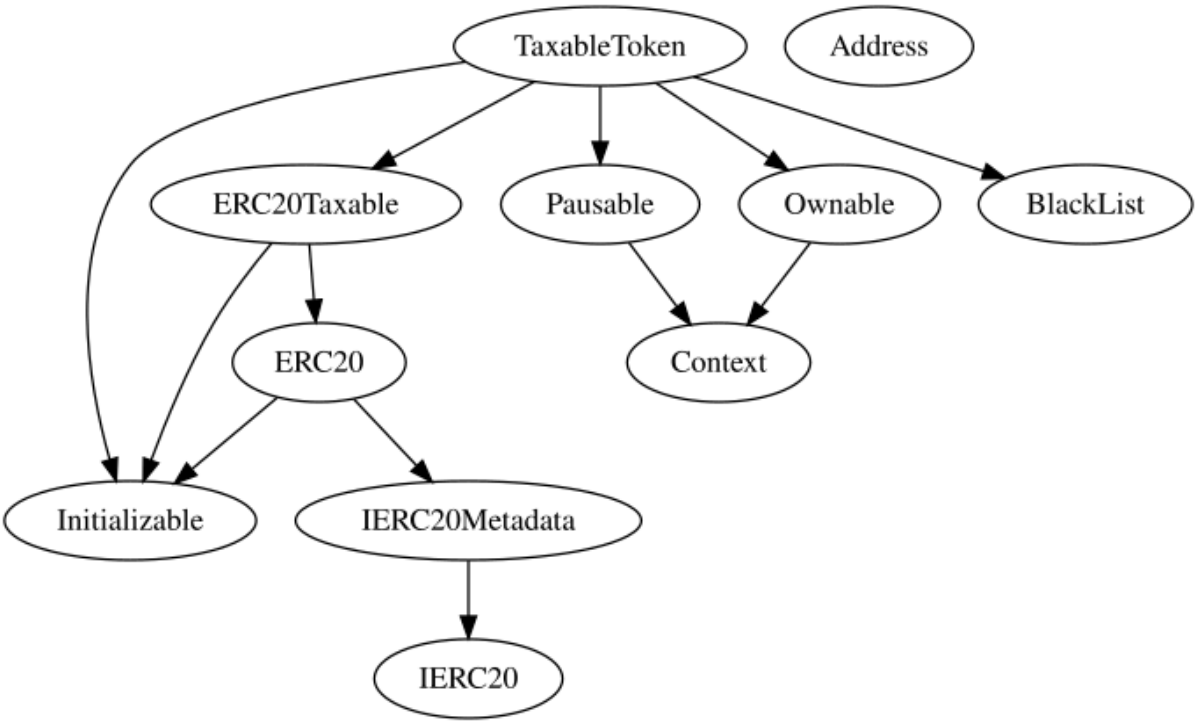
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

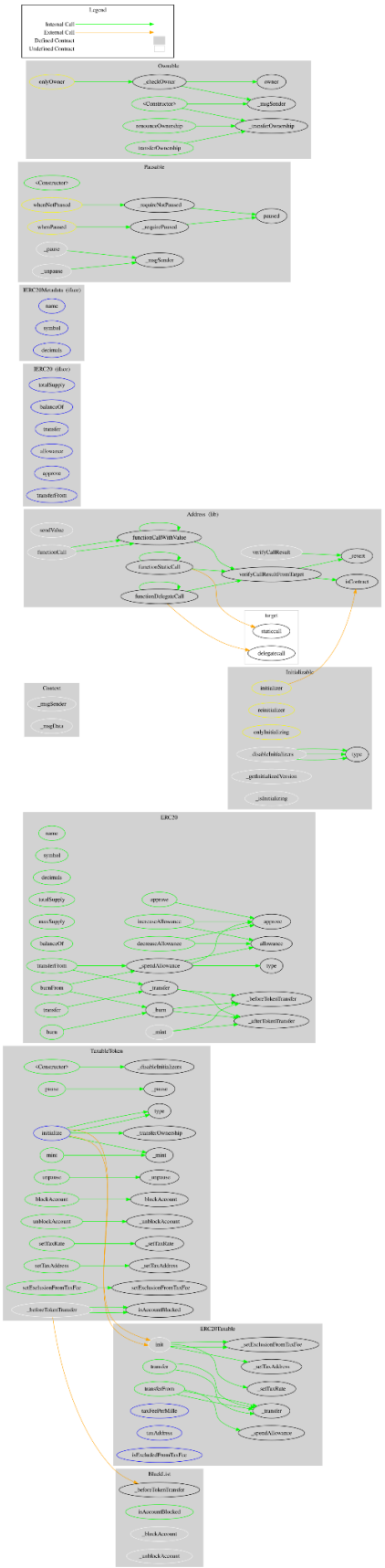
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
TaxableToken	Implementation	Initializable, ERC20Taxable, Pausable, Ownable, BlackList		
		Public	✓	-
	initialize	External	✓	initializer
	pause	Public	✓	onlyOwner
	unpause	Public	✓	onlyOwner
	mint	Public	✓	onlyOwner
	blockAccount	Public	✓	onlyOwner
	unblockAccount	Public	✓	onlyOwner
	setTaxRate	Public	✓	onlyOwner
	setTaxAddress	Public	✓	onlyOwner
	setExclusionFromTaxFee	Public	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	whenNotPaused
ERC20Taxable	Implementation	Initializable, ERC20		
	init	Internal	✓	onlyInitializing
	transfer	Public	✓	-

	transferFrom	Public	✓	-
	taxFeePerMille	External		-
	taxAddress	External		-
	isExcludedFromTaxFee	External		-
	_setTaxRate	Internal	✓	
	_setTaxAddress	Internal	✓	
	_setExclusionFromTaxFee	Internal	✓	
BlackList	Implementation			
	isAccountBlocked	Public		-
	_blockAccount	Internal	✓	
	_unblockAccount	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
Pausable
<Constructor>
whenNotPaused
_requireNotPaused
paused
whenPaused
_requirePaused
_pause
_unpause
_msgSender
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Address (lib)
sendValue
functionCallWithValue
functionCall
verifyCallResult
_revert
functionStaticCall
verifyCallResultFromTarget
isContract
functionDelegateCall
target
staticcall
delegatecall
Context
_msgSender
_msgData
Initializable
initializer
reinitializer
onlyInitializing
_disableInitializers
type
_getInitializedVersion
_isInitializing
ERC20
name
symbol
decimals
totalSupply
maxSupply
balanceOf
transferFrom
burnFrom
transfer
burn
approve
increaseAllowance
decreaseAllowance
_spendAllowance
_transfer
burn
_mint
approve
allowance
type
_beforeTokenTransfer
_afterTokenTransfer
TaxableToken
<Constructor>
pause
type
initialize
mint
unpause
blockAccount
unblockAccount
setTaxRate
setTaxAddress
setExclusionFromTaxFee
_beforeTokenTransfer
_disableInitializers
pause
type
_transferOwnership
_mint
_unpause
blockAccount
_unblockAccount
_setTaxRate
_setTaxAddress
setExclusionFromTaxFee
isAccountBlocked
ERC20Taxable
init
setExclusionFromTaxFee
setTaxAddress
setTaxRate
transferFrom
_transfer
_spendAllowance
taxFeePerMille
taxAddress
isExcludedFromTaxFee
BlockList
_beforeTokenTransfer
isAccountBlocked
_blockAccount
_unblockAccount

Summary

TrumpJr contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. TrumpJr 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.

Initial Audit, 5 Mar 2024

At the time of the audit report, the contract with address 0xe137407Dbf9a768f342DE7b88607e186fC9c09fB is pointed out by the following proxy address: 0x7fe0B79c1c17fe676223eb7f526B0038E6b419D6.

The ownership of the proxy contract has been renounced. The information regarding the transaction can be accessed through the following link: https://etherscan.io/tx/0x1f40546338f8e35e95f6ab15dece5bdacb9ed2589c962cef88eebcbc10f64f9b

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