Cyberscope

Audit Report

Immortl

January 2023

SHA256 d25af69d47d2a23397e87618ce61b2962636f739c363ed45dcd8f4ff20c33f55

Audited by © cyberscope

Table of Contents

Review

Contract Name	OneImmortl_ERC20_IMRTL
Testing Deploy	https://testnet.bscscan.com/address/0x9bae42af3cb50041544adb84baae39b0598165bc
Symbol	IMRTL
Decimals	18
Total Supply	400,000,000

Audit Updates

Initial Audit	25 Jan 2023
Corrected Phase 2	28 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/testingDeploy/OneImmortl_ERC20_IMRTL.sol	d25af69d47d2a23397e87618ce61b2962636f739c363ed45dcd8f4ff20c33f55

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	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	Unresolved
●	BC	Blacklists Addresses	Passed

BT - Burns Tokens

Criticality	Critical
Location	contracts/testingDeploy/OneImmortl_ERC20_IMRTL.sol#L433
Status	Unresolved

Description

Any user has the authority to burn tokens from any address. The users may take advantage of it by calling the `burn` function. As a result, the targeted address will lose the corresponding tokens.

```
function burn(address _from, uint256 _amount) external
{
    _burn(_from, _amount);
}
```

The following testing deployment depicts the burning functionality.

Token

https://testnet.bscscan.com/token/0x9bae42AF3cB50041544aDb84BAae39B0598165bc

1. The token is created and the tokens are transferred to `0x0f13e144de84b6ea420862b2fad1f9354ab1f8c6` https://testnet.bscscan.com/tx/0x85e63ee24fff68b264c106444dce5b34d00e9381c3f94b05845684bdd51cbbae
2. 1000 tokens are transferred to `0x9bae42af3cb50041544adb84baae39b0598165bc` https://testnet.bscscan.com/tx/0xfe696196f671a965221d4bc9a272f4c9dd9c56c8db0e7ee88c3574dbf0c50ae1
3. The address `0x0f13e144de84b6ea420862b2fad1f9354ab1f8c6` burns tokens from the address `0x9bae42af3cb50041544adb84baae39b0598165bc`

https://testnet.bscscan.com/tx/0x21af59ba36b375359dde13758f63c2c1fe54ef230752a098a2aa1ed8fbcf71f8

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/testingDeploy/OneImmortl_ERC20_IMRTL.sol#L8,156,163,198,210,217,224,235,240,270,280,290,313,320,321,322,433,442,462,490,514
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 OneImmortl_ERC20_IMRTL is
    Ownable
{
    //=======================
    // STRUCTS
    //=======================
...
        else if (_type == -1)
        {
            return maxTax.sell;
        }
        return maxTax.p2p;
    }
}

...
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/testingDeploy/OneImmortl_ERC20_IMRTL.sol#L166
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
taxWallet = _wallet
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/testingDeploy/OneImmortl_ERC20_IMRTL.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 ^0.8.17;
```

Recommendation

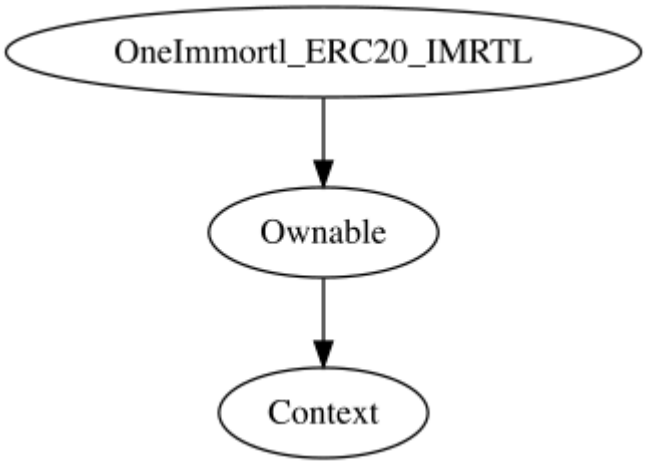
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest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

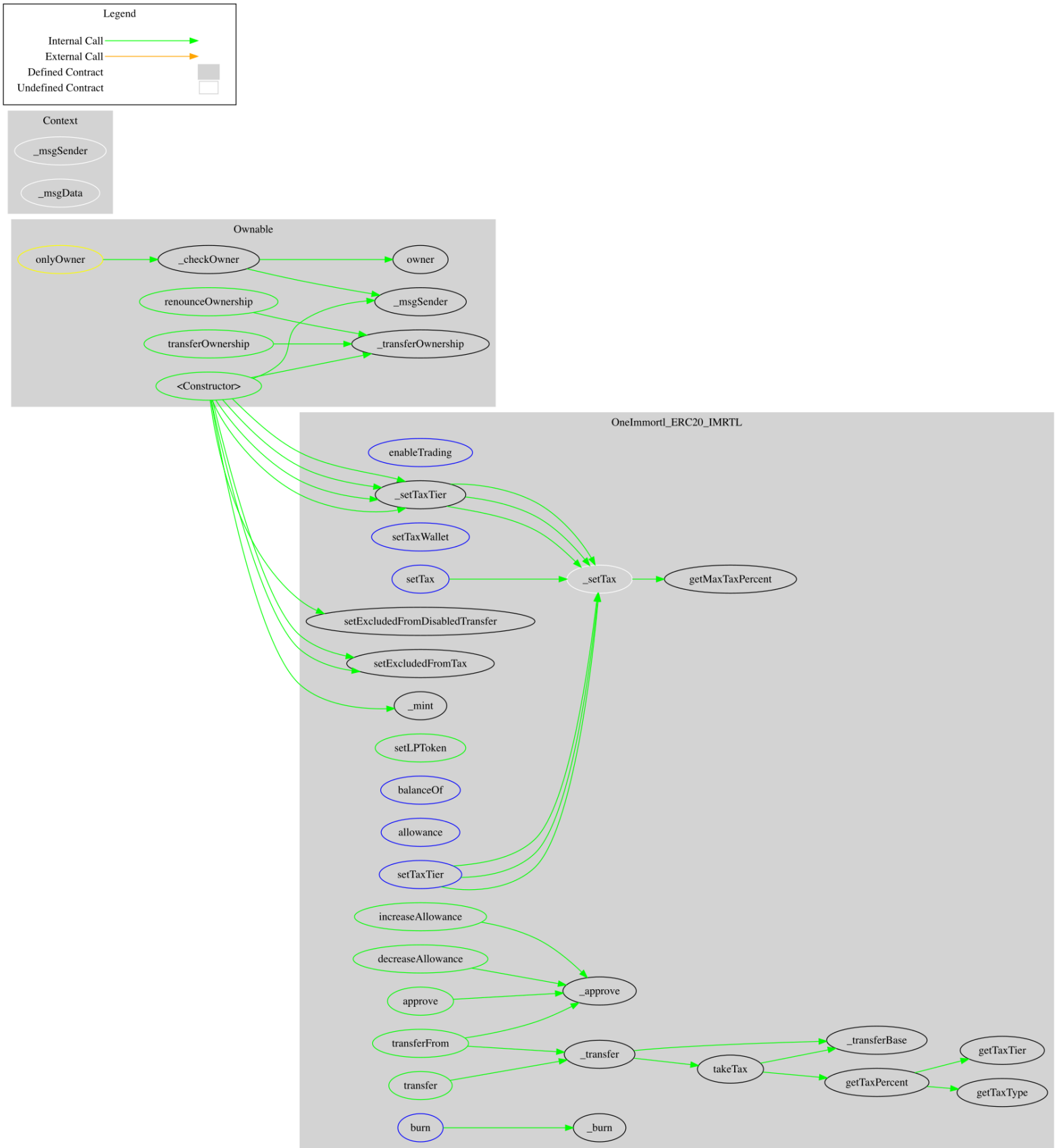
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
OneImmortl_ERC20_IMRTL	Implementation	Ownable		
		Public	✓	-
	enableTrading	External	✓	onlyOwner
	setExcludedFromDisabledTransfer	Public	✓	onlyOwner
	setTaxWallet	External	✓	onlyOwner
	_setTax	Internal	✓	onlyOwner
	setTax	External	✓	-
	_setTaxTier	Internal	✓	
	setTaxTier	External	✓	-
	setExcludedFromTax	Public	✓	onlyOwner
	setLPToken	Public	✓	onlyOwner

	balanceOf	External		-
	allowance	External		-
	_approve	Internal	✓	
	approve	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_transferBase	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	burn	External	✓	-
	takeTax	Internal	✓	
	getTaxType	Internal		
	getTaxTier	Internal		
	getTaxPercent	Public		-
	getMaxTaxPercent	Internal		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Context
_msgSender
_msgData
Ownable
onlyOwner
_checkOwner
owner
renounceOwnership
_msgSender
transferOwnership
_transferOwnership
<Constructor>
OneImmortl_ERC20_IMRTL
enableTrading
_setTaxTier
setTaxWallet
setTax
_setTax
getMaxTaxPercent
setExcludedFromDisabledTransfer
setExcludedFromTax
_mint
setLPToken
balanceOf
allowance
setTaxTier
increaseAllowance
decreaseAllowance
_approve
approve
transferFrom
_transfer
_transferBase
takeTax
getTaxTier
transfer
getTaxPercent
getTaxType
burn
_burn

Summary

Any user can burn the tokens from any other user. if the users abuse the burning functionality, then the contract could lose its tokens and the total supply will tend to zero.

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