



Smart contracts security assessment

Final report

Tariff: Standard

Cybercash Token

November 2021



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	3
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	5
7. Disclaimer	8
8. Static code analysis result	9

Introduction

The report has been prepared for the Cybercash Token team.

ERC20 token with minting open for the owner and limited by maxSupply immutable variable, set about 1000 times higher than the initial token supply.

Name	Cybercash Token
Audit date	2021-11-11 - 2021-11-15
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
CybercashToken	0xD36Bb7849a8c50b509baE872F44d5C1C7DC0e96C
TokenAccessControl	0xD36Bb7849a8c50b509baE872F44d5C1C7DC0e96C
SafeMath	0xD36Bb7849a8c50b509baE872F44d5C1C7DC0e96C

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities

- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	not passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	not passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	not passed
State Variable Default Visibility	passed
Reentrancy	passed

Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

1. Name and symbol changing (CybercashToken)

The name and symbol of the token could be changed by the owner.

Recommendation: Users should pay attention to not being involved in fraudulent actions with mimic tokens.

2. Open minting (CybercashToken)

Open for owner minting could be dangerous if the owner gets hacked or acts maliciously.

Recommendation: Token ownership should be secured by transferring to the special contract, i.g. a multisig or Timelock contract.

Low severity issues

1. Lack of error messages (CybercashToken)

All the require() statements lack corresponding error messages.

Recommendation: Returning a specific revert reason helps users interact with the contract.

2. Using of assert() (CybercashToken)

assert() usage should be deprecated in favor of require(), as the latter saves gas and returns the error message.

Recommendation: We recommend using the assert() only for the conditions that never fail just as a precaution.

3. Lack of events (CybercashToken)

The mint() function should emit a corresponding event.

Recommendation: A standard minting event is a transfer() from zero address to the recipient.

4. Troublesome renouncing (TokenAccessControl)

Ownership renouncing needs a specific proxy contract to be deployed.

Recommendation: To renounce the Cybercash token ownership, one should transfer it to a contract with implemented acceptOwnership() ability. That contract also must not be able to transfer it further.

5. Lack of error messages (TokenAccessControl)

All the require() statements lack corresponding error messages.

Recommendation: Returning a specific revert reason helps users interact with the contract.

6. Using of assert() (SafeMath)

assert() usage should be deprecated in favor of require(), as the latter saves gas and returns the error message.

Recommendation: We recommend using the assert() only for the conditions that never fail just as a precaution.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard 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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

Static code analysis result

Slither tool analysis results:

CybercashToken.mint(uint256) (contracts/CybercashToken.sol#130-135) should emit an event for:

- totalSupply = totalSupply.add(_amount) (contracts/CybercashToken.sol#133)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

SafeMath.div(uint256,uint256) (contracts/CybercashToken.sol#88-90) is never used and should be removed

SafeMath.mul(uint256,uint256) (contracts/CybercashToken.sol#79-86) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

Pragma version>=0.7.0<0.9.0 (contracts/CybercashToken.sol#6) is too complex
solc-0.8.4 is not recommended for deployment

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Parameter TokenAccessControl.transferOwnership(address)._newOwner (contracts/CybercashToken.sol#56) is not in mixedCase

Parameter TokenAccessControl.setPause(bool)._paused (contracts/CybercashToken.sol#68) is not in mixedCase

Parameter CybercashToken.changeNameAndSymbol(string,string)._name (contracts/CybercashToken.sol#125) is not in mixedCase

Parameter CybercashToken.changeNameAndSymbol(string,string)._symbol (contracts/CybercashToken.sol#125) is not in mixedCase

Parameter CybercashToken.mint(uint256)._amount (contracts/CybercashToken.sol#130) is not in mixedCase

Parameter CybercashToken.balanceOf(address)._owner (contracts/CybercashToken.sol#137) is not in mixedCase

Parameter CybercashToken.transfer(address,uint256)._to (contracts/CybercashToken.sol#141) is not in mixedCase

Parameter CybercashToken.transfer(address,uint256)._value (contracts/CybercashToken.sol#141) is not in mixedCase

Parameter CybercashToken.batchTransfer(address[],uint256[])._to (contracts/CybercashToken.sol#146) is not in mixedCase

Parameter CybercashToken.batchTransfer(address[],uint256[])._value (contracts/CybercashToken.sol#146) is not in mixedCase

Parameter CybercashToken.transferFrom(address,address,uint256)._from (contracts/CybercashToken.sol#155) is not in mixedCase

Parameter CybercashToken.transferFrom(address,address,uint256)._to (contracts/CybercashToken.sol#155) is not in mixedCase

Parameter CybercashToken.transferFrom(address,address,uint256)._value (contracts/CybercashToken.sol#155) is not in mixedCase

Parameter CybercashToken.approve(address,uint256)._spender (contracts/CybercashToken.sol#162) is not in mixedCase

Parameter CybercashToken.approve(address,uint256)._value (contracts/CybercashToken.sol#162) is not in mixedCase

Parameter CybercashToken.allowance(address,address)._owner (contracts/CybercashToken.sol#168) is not in mixedCase

Parameter CybercashToken.allowance(address,address)._spender (contracts/CybercashToken.sol#168) is not in mixedCase

Parameter CybercashToken.approveAndCall(address,uint256,bytes)._spender (contracts/CybercashToken.sol#172) is not in mixedCase

Parameter CybercashToken.approveAndCall(address,uint256,bytes)._value (contracts/CybercashToken.sol#172) is not in mixedCase

Parameter CybercashToken.approveAndCall(address,uint256,bytes)._extraData (contracts/CybercashToken.sol#172) is not in mixedCase

Parameter CybercashToken.burn(uint256)._value (contracts/CybercashToken.sol#179) is not in mixedCase

Parameter CybercashToken.burnFrom(address,uint256)._from (contracts/CybercashToken.sol#187) is not in mixedCase

Parameter CybercashToken.burnFrom(address,uint256)._value (contracts/CybercashToken.sol#187) is not in mixedCase

Parameter CybercashToken.withdrawBalance(uint256)._amount (contracts/CybercashToken.sol#216) is not in mixedCase

Variable CybercashToken._balances (contracts/CybercashToken.sol#110) is not in mixedCase

Variable CybercashToken._allowed (contracts/CybercashToken.sol#111) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

CybercashToken.constructor() (contracts/CybercashToken.sol#114-121) uses literals with too many digits:

- maxSupply = 1000000000000 * 10 ** uint256(decimals) (contracts/CybercashToken.sol#119)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits>

balanceOf(address) should be declared external:

- CybercashToken.balanceOf(address) (contracts/CybercashToken.sol#137-139)
- ERC20Interface.balanceOf(address) (contracts/CybercashToken.sol#24)

transfer(address,uint256) should be declared external:

- CybercashToken.transfer(address,uint256) (contracts/CybercashToken.sol#141-144)
- ERC20Interface.transfer(address,uint256) (contracts/CybercashToken.sol#25)

transferFrom(address,address,uint256) should be declared external:

- CybercashToken.transferFrom(address,address,uint256) (contracts/CybercashToken.sol#155-160)
- ERC20Interface.transferFrom(address,address,uint256) (contracts/CybercashToken.sol#26)

allowance(address,address) should be declared external:

- CybercashToken.allowance(address,address) (contracts/CybercashToken.sol#168-170)
- ERC20Interface.allowance(address,address) (contracts/CybercashToken.sol#28)

setPause(bool) should be declared external:

- TokenAccessControl.setPause(bool) (contracts/CybercashToken.sol#68-73)

receiveApproval(address,uint256,address,bytes) should be declared external:

- TokenRecipient.receiveApproval(address,uint256,address,bytes) (contracts/CybercashToken.sol#105)

changeNameAndSymbol(string,string) should be declared external:

- CybercashToken.changeNameAndSymbol(string,string) (contracts/CybercashToken.sol#125-128)

mint(uint256) should be declared external:

- CybercashToken.mint(uint256) (contracts/CybercashToken.sol#130-135)

batchTransfer(address[],uint256[]) should be declared external:

- CybercashToken.batchTransfer(address[],uint256[]) (contracts/CybercashToken.sol#146-153)

approveAndCall(address,uint256,bytes) should be declared external:

- CybercashToken.approveAndCall(address,uint256,bytes) (contracts/CybercashToken.sol#172-177)

burn(uint256) should be declared external:

- CybercashToken.burn(uint256) (contracts/CybercashToken.sol#179-185)

burnFrom(address,uint256) should be declared external:

- CybercashToken.burnFrom(address,uint256) (contracts/CybercashToken.sol#187-195)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

. analyzed (5 contracts with 75 detectors), 44 result(s) found

