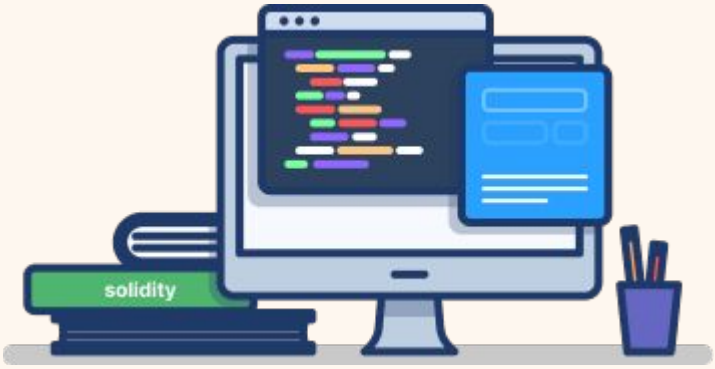
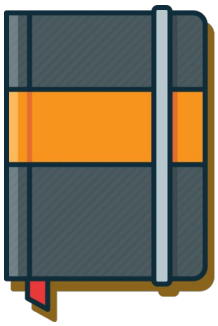
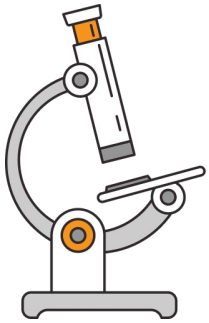
Coinbae Audit

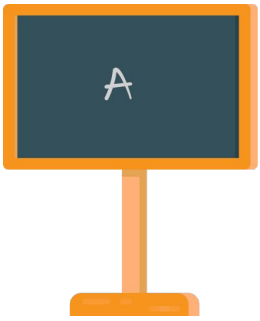


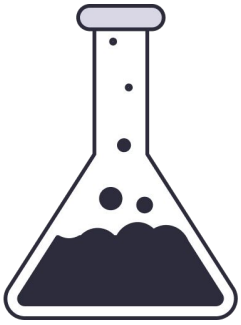
Blockchain Cuties March 2021

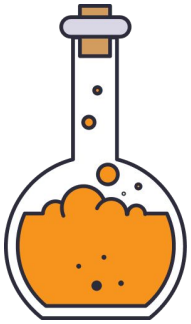
Contents











Introduction

Audit:

In March 2021 Coinbae's audit report division performed an audit for the Blockchain Cuties Universe Governance Token Contract.

BCUG_flat.sol

Blockchain Cuties Universe:

Blockchain Cuties Universe will become the first DeFi game that leverages the power of five blockchains. The game's team believes that the synergy of NFT and DeFi technologies will propel the game's economy and engagement to new heights.

Scope of the audit:

The following Coinbae audit will cover assertions and property checking, ERC Standards, solidity coding best practices, conformity to the solidity style guide.

Overview:

Name: Blockchain Cuties Universe Governance Token Contract
Website: https://www.blockchaincuties.finance/
Contract's purpose is minting and distribution of BCUG ERC-20 token.
Current total issuance: 10'000'000 BCUG.

Audit Report Scope

Business logic testing:

The Coinbae audit team was asked to confirm the following:

1. Contract's purpose is minting and distribution of BCUG ERC-20 token.

2. Current total issuance should be 1'000'000 BCUG (that might get changed to 10'000'000 before the launch. Anyway, this number should be set in stone before the launch and could not be altered after.

3. All important actions, such as minting of tokens, transfer of tokens, freezing of tokens/all meaningful operations regarding the token should be used ONLY with MULTISIG. There's going to be a total of 5 addresses eligible for MULTISIG and each transaction must require any 3 of those 5 in order to start.

4. There shouldn't be any important functions on this token contract that can be executed by one person solely.

5. Contract should have the ability to freeze any token transfer action in case some exchange where the token is listed gets hacked (like it happened during KuCoin hack in 2020) and ability to reimburse hacked tokens one way or another.

6. Incase someone mistakenly sent other ERC20 tokens to the contract, or ETH, it should be possible to transfer those funds to a different address, also using Multisig.

Audit Report Scope

Assertions and Property Checking:

1. Solidity assert violation.
2. Solidity AssertionFailed event.

ERC Standards:

1. Incorrect ERC20 implementation.

Solidity Coding Best Practices:

1. Outdated compiler version.
2. No or floating compiler version set.
3. Use of right-to-left-override control character.
4. Shadowing of built-in symbol.
5. Incorrect constructor name.
6. State variable shadows another state variable.
7. Local variable shadows a state variable.
8. Function parameter shadows a state variable.
9. Named return value shadows a state variable.
10. Unary operation without effect Solidity code analysis.
11. Unary operation directly after assignment.
12. Unused state variable.
13. Unused local variable.
14. Function visibility is not set.
15. State variable visibility is not set.

Solidity Coding Best Practices (Continued):

16. Use of deprecated functions: call code(), sha3(), ...
17. Use of deprecated global variables (msg.gas, ...).
18. Use of deprecated keywords (throw, var).
19. Incorrect function state mutability.
20. Does the code conform to the Solidity styleguide.

Convert code to conform Solidity style guide:

1. Convert all code so that it is structured accordingly the Solidity style guide.

Audit Report Scope

Categories:

High Severity:

High severity issues opens the contract up for exploitation from malicious actors. We do not recommend deploying contracts with high severity issues.

Medium Severity Issues:

Medium severity issues are errors found in contracts that hampers the effectiveness of the contract and may cause outcomes when interacting with the contract. It is still recommended to fix these issues.

Low Severity Issues:

Low severity issues are warning of minor impact on the overall integrity of the contract. These can be fixed with less urgency.

Optimization Issues:

Optimization issues are issues that pose no security risk or expose any underlying vulnerabilities, but instead make the contract more efficient.

Informational Issues:

Informational issues are issues that point to smart contract coding best practises.

Audit Report

33	33	0
Identified	Confirmed	Critical

0	0	0
High	Medium	Low

Optimization issues identified:	**18**
Informational issues identified:	**15**

Analysis:
BCUG_flat.sol

Risk:
Low

Optimization issues identified:

Constable-states

GovernanceToken.nonce (BCUG_flat.sol#1051) should be constant

External-function

name() should be declared external:
- ERC20.name() (BCUG_flat.sol#172-174)

symbol() should be declared external:
- ERC20.symbol() (BCUG_flat.sol#180-182)

decimals() should be declared external:
- ERC20.decimals() (BCUG_flat.sol#197-199)

balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (BCUG_flat.sol#211-213)

transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (BCUG_flat.sol#223-226)

transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (BCUG_flat.sol#223-226)

transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (BCUG_flat.sol#260-268)

increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (BCUG_flat.sol#282-285)

decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (BCUG_flat.sol#301-307)

Optimization issues identified:

External-function

isSigner(address) should be declared external:
- AccessControl.isSigner(address) (BCUG_flat.sol#793-795)

isFrozen(address) should be declared external:
- AccessControl.isFrozen(address) (BCUG_flat.sol#800-802)

burn(uint256) should be declared external:
- ERC20Burnable.burn(uint256) (BCUG_flat.sol#866-868)

burnFrom(address,uint256) should be declared external:
- ERC20Burnable.burnFrom(address,uint256) (BCUG_flat.sol#881-886)

getOperationHash(bytes32,address,uint256) should be declared external:
- GovernanceToken.getOperationHash(bytes32,address,uint256) (BCUG_flat.sol#1178-1180)

transferWithData(address,uint256,bytes) should be declared external:
- BCUG.transferWithData(address,uint256,bytes) (BCUG_flat.sol#1298-1305)

transferToContract(address,uint256,bytes) should be declared external:
- BCUG.transferToContract(address,uint256,bytes) (BCUG_flat.sol#1308-1314)

transferToAddress(address,uint256,bytes) should be declared external:
- BCUG.transferToAddress(address,uint256,bytes) (BCUG_flat.sol#1327-1331)

Informational issues identified:

Assembly

ECDSA.recover(bytes32,bytes) (BCUG_flat.sol#688-709) uses assembly
- INLINE ASM (BCUG_flat.sol#702-706)

Solc-version

Pragma version^0.8.0 (BCUG_flat.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11.

solc-0.8.0 is not recommended for deployment .

naming-convention

Parameter AccessControl.isSigner(address)._addr (BCUG_flat.sol#793) is not in mixedCase

Unused-state

ERC20._balances (BCUG_flat.sol#146) is never used in BCUG (BCUG_flat.sol#1246-1332)

ERC20._allowances (BCUG_flat.sol#148) is never used in BCUG (BCUG_flat.sol#1246-1332)

ERC20._totalSupply (BCUG_flat.sol#150) is never used in BCUG (BCUG_flat.sol#1246-1332)

AccessControl._frozen (BCUG_flat.sol#760) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.MINT_TYPEHASH (BCUG_flat.sol#1031) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.MINT_BULK_TYPEHASH (BCUG_flat.sol#1034) is never used in BCUG (BCUG_flat.sol#1246-1332)

Informational issues identified:

Unused-state

GovernanceToken.FREEZE_TYPEHASH (BCUG_flat.sol#1037) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.FREEZE_BULK_TYPEHASH (BCUG_flat.sol#1040) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.BURN_FROZEN_TYPEHASH (BCUG_flat.sol#1043) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.FREEZE_AND_BURN_TYPEHASH (BCUG_flat.sol#1046) is never used in BCUG (BCUG_flat.sol#1246-1332)

GovernanceToken.PAUSE_TYPEHASH (BCUG_flat.sol#1049) is never used in BCUG (BCUG_flat.sol#1246-1332)

Business logic issues:

The Coinbae audit team was asked to confirm the following:

1. Contract's purpose is minting and distribution of BCUG ERC-20 token.
   Coinbae comment: Confirmed

2. Current total issuance should be 1'000'000 BCUG (that might get changed to 10'000'000 before the launch.
   Coinbae comment: The current total contained in this contract is 10,000,000 BCUG. However since the audited contract has not been deployed yet, the validity thieroff on deployment cannot be authenticated yet.

3. All important actions, such as minting of tokens, transfer of tokens, freezing of tokens/all meaningful operations regarding the token should be used ONLY with MULTISIG. There's going to be a total of 5 addresses eligible for MULTISIG and each transaction must require any 3 of those 5 in order to start.
   Coinbae comment: Confirmed, the only function where less than three signers are needed is when ERC20 tokens are mistakenly sent to the contract address, which doesn't pose a security risk to BUCG token holders.

4. There shouldn't be any important functions on this token contract that can be executed by one person solely.
   Coinbae comment: Confirmed, the only function where less than three signers are needed is when ERC20 tokens are mistakenly sent to the contract address, which doesn't pose a security risk to BUCG token holders.

Business logic issues:

The Coinbae audit team was asked to confirm the following:

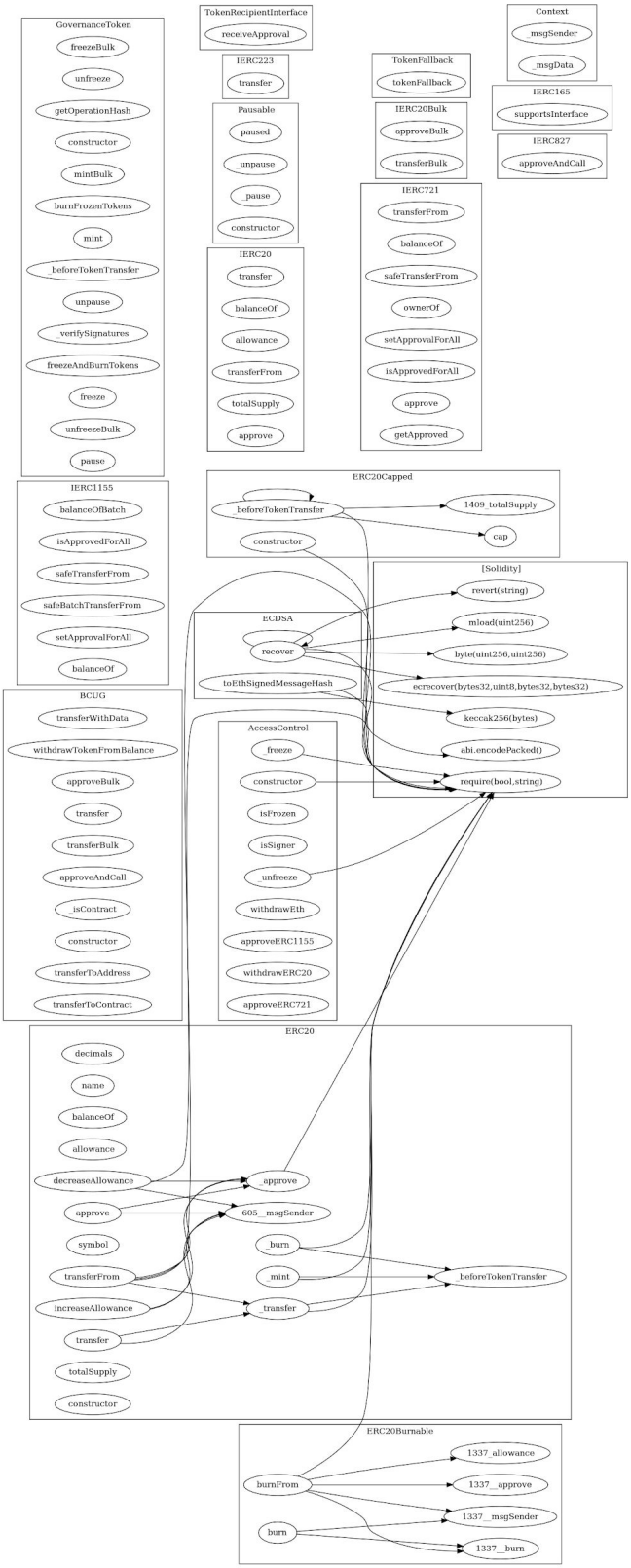
5. Contract should have the ability to freeze any token transfer action in case some exchange where the token is listed gets hacked (like it happened during KuCoin hack in 2020) and ability to reimburse hacked tokens one way or another.
   Coinbae comment: These functions exist and to execute the "pause", "unpause", "freeze", "unfreeze", "freezAndBurn" and "minting" functions three signers are needed.

6. Incase someone mistakenly sent other ERC20 tokens to the contract, or ETH, it should be possible to transfer those funds to a different address, also using Multisig.
   Coinbae comment: Note that coins sent mistakenly to a contract address can be retrieved. In this case a single multisig user can withdraw. Other ERC20 tokens sent mistakenly to the contract and then to be sent back does not pose a security risk to BCUG and cannot be exploited.

Conclusion:

Coinbae deems this contract to be highly efficient and safe to interact with. There are no known exploits or malevolent code in the contract.

Contract Flow



Audit Team

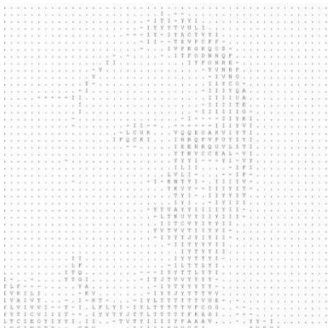
Team Lead: Eelko Neven

Eelko has been in the it/security space since 1991. His passion started when he was confronted with a formatted hard drive and no tools to undo it. At that point he started reading a lot of material on how computers work and how to make them work for others. After struggling for a few weeks he finally wrote his first HD data recovery program. Ever since then when he was faced with a challenge he just persisted until he had a solution.

This mindset helped him tremendously in the security space. He found several vulnerabilities in large corporation servers and notified these corporations in a responsible manner. Among those are Google, Twitter, General Electrics etc.

For the last 12 years he has been working as a professional security /code auditor and performed over 1500 security audits / code reviews, he also wrote a similar amount of reports.

He has extensive knowledge of the Solidity programming language and this is why he loves to do Defi and other smartcontract reviews.



Email:
info@coinbae.com

Disclaimer

Coinbae audit is not a security warranty, investment advice, or an endorsement of the Blockchain Cuties. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Blockchain Cuties team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.