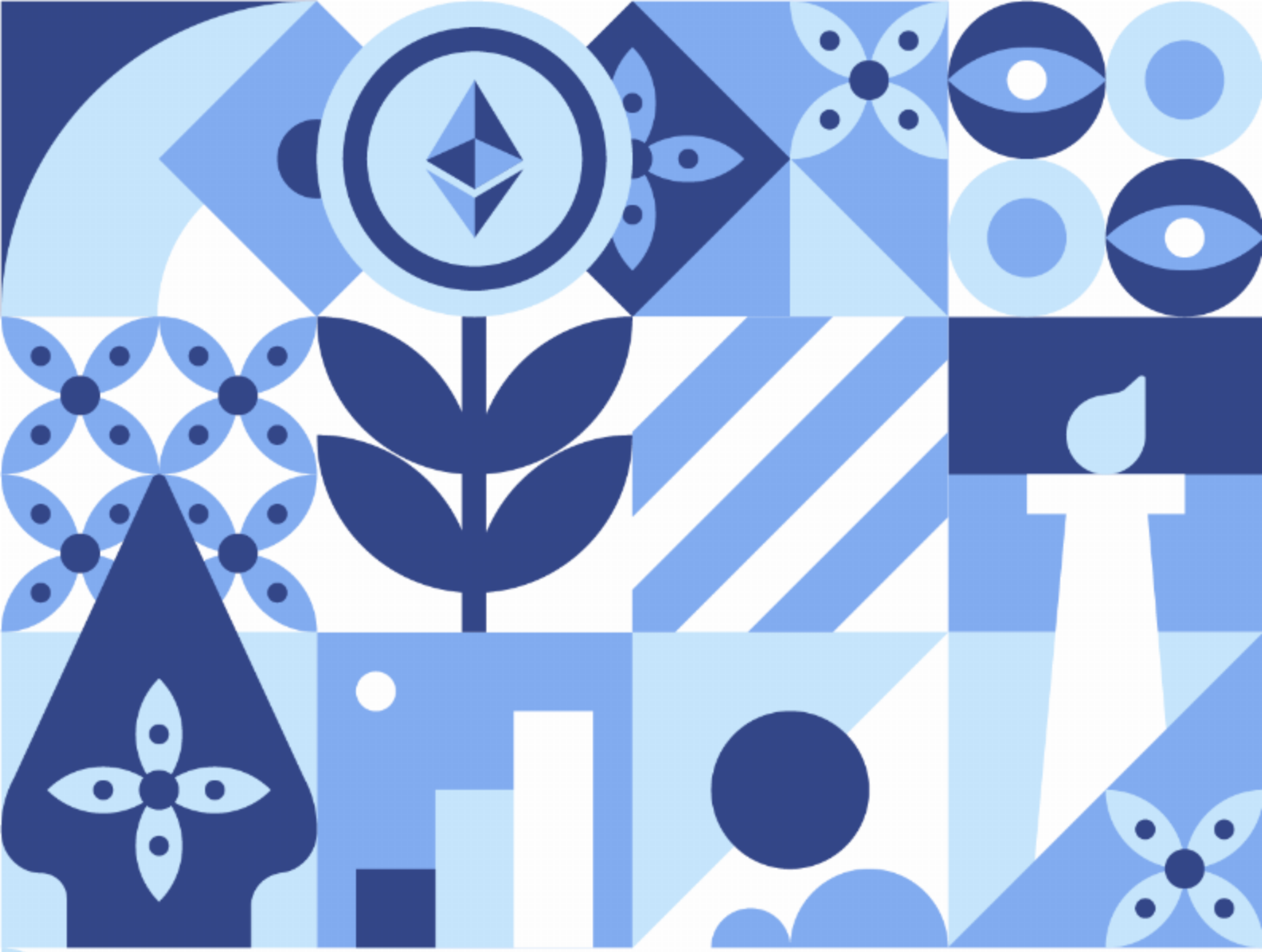

Blockchain Cuties Universe Governance Token

Smart Contract Audit Report

SYSFIXED

15 Mar 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Blockchain Cuties Universe Governance Token	BCUG	Ethereum

Addresses

Contract address	0x14da7b27b2e0fedefe0a664118b0c9bc68e2e9af
Contract deployer address	0x9D056B46cBa330adAe643FBb57F0a47b1155Ce56

Project Website

https://blockchaincuties.com/

Codebase

https://etherscan.io/address/0x14da7b27b2e0fedefe0a664118b0c9bc68e2e9af#code

SUMMARY

Blockchain Cuties is a new collectible crypto game with adventures where you get to play with puppies, lizards, bear cubs, cats and other real and fantasy creatures alike. Each cutie is unique and 100% belongs to you. You get to collect them, breed them, test their skills in battles, arm them and even level them up! In-game economy lets you trade cuties using smart contracts on Ethereum, EOS, TRON and NEO blockchains. Each cutie, which is a non-fungible, can be transferred or sold to other players just like a regular cryptocurrency.

Contract Summary

Documentation Quality

Blockchain Cuties Universe Governance Token provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Blockchain Cuties Universe Governance Token with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 761.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 269, 287, 308, 335, 336, 355, 356, 378, 379, 779, 888, 926, 1085, 1116, 1127, 1210, 1214, 1218, 1222, 1228, 1280 and 1288.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 780, 781, 782, 1082, 1086, 1086, 1113, 1117, 1124, 1128, 1211, 1215, 1219, 1222, 1282, 1282, 1290 and 1290.

CONCLUSION

We have audited the Blockchain Cuties Universe Governance Token project released on March 2021 to discover issues and identify potential security vulnerabilities in Blockchain Cuties Universe Governance Token Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the Blockchain Cuties Universe Governance Token smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Sunday Mar 14 2021 22:14:31 GMT+0000 (Coordinated Universal Time)
Finished	Monday Mar 15 2021 12:46:29 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BCUG.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 269

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
268   require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
269   _approve(sender, _msgSender(), currentAllowance - amount);
270
271   return true;
272   }
273
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 287

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
286   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
287   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
288   return true;
289   }
290
291
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 308

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
307   require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
308   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
309
310   return true;
311   }
312
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 335

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
334   require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
335   _balances[sender] = senderBalance - amount;
336   _balances[recipient] += amount;
337
338   emit Transfer(sender, recipient, amount);
339
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 336

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
335   _balances[sender] = senderBalance - amount;
336   _balances[recipient] += amount;
337
338   emit Transfer(sender, recipient, amount);
339   }
340
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 355

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
354
355   _totalSupply += amount;
356   _balances[account] += amount;
357   emit Transfer(address(0), account, amount);
358   }
359
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 356

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
355   _totalSupply += amount;
356   _balances[account] += amount;
357   emit Transfer(address(0), account, amount);
358   }
359
360
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 378

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
377   require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
378   _balances[account] = accountBalance - amount;
379   _totalSupply -= amount;
380
381   emit Transfer(account, address(0), amount);
382
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 379

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
378   _balances[account] = accountBalance - amount;
379   _totalSupply -= amount;
380
381   emit Transfer(account, address(0), amount);
382   }
383
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 779

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
778
779   for (uint8 i = 0; i < allowedSigners.length; i++) {
780   require(allowedSigners[i] != address(0), "AccessControl: Invalid signer address");
781   require(!signers[allowedSigners[i]], "AccessControl: Signer address duplication");
782   signers[allowedSigners[i]] = true;
783
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 888

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
887   require(currentAllowance >= amount, "BCUG: burn amount exceeds allowance");
888   _approve(account, _msgSender(), currentAllowance - amount);
889   _burn(account, amount);
890   }
891   }
892
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 926

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
925   if (from == address(0)) { // When minting tokens
926   require(totalSupply() + amount <= cap(), "ERC20Capped: cap exceeded");
927   }
928   }
929   }
930
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1085

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1084
1085   for (uint i = 0; i < target.length; i++) {
1086   _mint(target[i], amount[i]);
1087   }
1088   }
1089
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1116

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1115
1116   for (uint i = 0; i < target.length; i++) {
1117   _freeze(target[i]);
1118   }
1119   }
1120
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1127

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1126
1127   for (uint i = 0; i < target.length; i++) {
1128   _unfreeze(target[i]);
1129   }
1130   }
1131
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1210

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1209
1210   address[] memory recovered = new address[](signatures.length + 1);
1211   recovered[0] = msg.sender;
1212   emit SignedBy(msg.sender);
1213
1214
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1214

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1213
1214   for (uint i = 0; i < signatures.length; i++) {
1215   address addr = operationHash.recover(signatures[i]);
1216   require(isSigner(addr), "AccessControl: recovered address is not signer");
1217
1218
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1218

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1217
1218   for (uint j = 0; j < recovered.length; j++) {
1219   require(recovered[j] != addr, "AccessControl: signer address used more than
once");
1220   }
1221
1222
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1222

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1221
1222   recovered[i + 1] = addr;
1223   emit SignedBy(addr);
1224   }
1225
1226
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1228

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1227
1228   nonce++;
1229   }
1230   }
1231
1232
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1280

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1279   require(to.length == tokens.length, "transferBulk: to.length != tokens.length");
1280   for (uint i = 0; i < to.length; i++)
1281   {
1282   _transfer(msg.sender, to[i], tokens[i]);
1283   }
1284
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1288

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BCUG.sol

Locations

```
1287   require(spender.length == tokens.length, "approveBulk: spender.length !=
tokens.length");
1288   for (uint i = 0; i < spender.length; i++)
1289   {
1290   _approve(msg.sender, spender[i], tokens[i]);
1291   }
1292
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- BCUG.sol

Locations

```
6
7   pragma solidity ^0.8.0;
8
9   /**
10   * @dev Interface of the ERC165 standard, as defined in the
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 761

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "signers" is internal. Other possible visibility settings are public and private.

Source File

- BCUG.sol

Locations

```
760   // The addresses that can co-sign transactions on the wallet
761   mapping(address => bool) signers;
762
763   // Frozen account that cant move funds
764   mapping (address => bool) private _frozen;
765
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 780

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
779   for (uint8 i = 0; i < allowedSigners.length; i++) {
780   require(allowedSigners[i] != address(0), "AccessControl: Invalid signer address");
781   require(!signers[allowedSigners[i]], "AccessControl: Signer address duplication");
782   signers[allowedSigners[i]] = true;
783   }
784
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 781

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
780   require(allowedSigners[i] != address(0), "AccessControl: Invalid signer address");
781   require(!signers[allowedSigners[i]], "AccessControl: Signer address duplication");
782   signers[allowedSigners[i]] = true;
783   }
784   }
785
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 782

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
781   require(!signers[allowedSigners[i]], "AccessControl: Signer address duplication");
782   signers[allowedSigners[i]] = true;
783   }
784   }
785
786
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1082

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1081
1082   bytes32 operationHash = getOperationHash(MINT_BULK_TYPEHASH, target[0],
target.length).toEthSignedMessageHash();
1083   _verifySignatures(signatures, operationHash);
1084
1085   for (uint i = 0; i < target.length; i++) {
1086
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1086

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1085   for (uint i = 0; i < target.length; i++) {
1086   _mint(target[i], amount[i]);
1087   }
1088   }
1089
1090
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1086

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1085   for (uint i = 0; i < target.length; i++) {
1086   _mint(target[i], amount[i]);
1087   }
1088   }
1089
1090
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1113

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1112
1113   bytes32 operationHash = getOperationHash(FREEZE_BULK_TYPEHASH, target[0],
target.length).toEthSignedMessageHash();
1114   _verifySignatures(signatures, operationHash);
1115
1116   for (uint i = 0; i < target.length; i++) {
1117
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1117

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1116   for (uint i = 0; i < target.length; i++) {
1117   _freeze(target[i]);
1118   }
1119   }
1120
1121
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1124

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1123
1124   bytes32 operationHash = getOperationHash(FREEZE_BULK_TYPEHASH, target[0],
target.length).toEthSignedMessageHash();
1125   _verifySignatures(signatures, operationHash);
1126
1127   for (uint i = 0; i < target.length; i++) {
1128
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1128

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1127   for (uint i = 0; i < target.length; i++) {
1128   _unfreeze(target[i]);
1129   }
1130   }
1131
1132
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1211

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1210   address[] memory recovered = new address[](signatures.length + 1);
1211   recovered[0] = msg.sender;
1212   emit SignedBy(msg.sender);
1213
1214   for (uint i = 0; i < signatures.length; i++) {
1215
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1215

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1214   for (uint i = 0; i < signatures.length; i++) {
1215   address addr = operationHash.recover(signatures[i]);
1216   require(isSigner(addr), "AccessControl: recovered address is not signer");
1217
1218   for (uint j = 0; j < recovered.length; j++) {
1219
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1219

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1218   for (uint j = 0; j < recovered.length; j++) {
1219   require(recovered[j] != addr, "AccessControl: signer address used more than
once");
1220   }
1221
1222   recovered[i + 1] = addr;
1223
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1222

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1221
1222   recovered[i + 1] = addr;
1223   emit SignedBy(addr);
1224   }
1225
1226
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1282

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1281   {
1282   _transfer(msg.sender, to[i], tokens[i]);
1283   }
1284   }
1285
1286
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1282

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1281   {
1282   _transfer(msg.sender, to[i], tokens[i]);
1283   }
1284   }
1285
1286
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1290

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1289   {
1290   _approve(msg.sender, spender[i], tokens[i]);
1291   }
1292   }
1293
1294
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1290

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BCUG.sol

Locations

```
1289   {
1290   _approve(msg.sender, spender[i], tokens[i]);
1291   }
1292   }
1293
1294
```

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

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 Sysfixed 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.