verichains

SECURITY AUDIT OF

CYBALL SMART CONTRACTS

Public Report

Nov 05, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Nov 05, 2021. We would like to thank the CyBall for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Cyball Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the application, along with some recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY 5

1.1. About Cyball Smart Contracts 5

1.2. Audit scope 5

1.3. Audit methodology 5

1.4. Disclaimer 6

2. AUDIT RESULT 7

2.1. Overview 7

2.2. Findings 7

2.2.1. CyBlocPackSale.sol - Reuse signature in buy function LOW 7

2.2.2. CyBlocPack.sol - Unsafe packs opening function LOW 9

2.3. Additional notes and recommendations 10

2.3.1. CyballMentorManager.sol - Useless code in finishMentor function INFORMATIVE 10

2.3.2. CyballMentorManager.sol - Useless code in startMentor function INFORMATIVE 11

2.3.3. CyBlocPack.sol - Unused SafeMath, SafeERC20, Ownable INFORMATIVE 13

2.3.4. CyBlocPack.sol - Inaccurate require statement message INFORMATIVE 13

2.3.5. CyBlocPack.sol - Reuse signature INFORMATIVE 13

2.3.6. CyBlocPackSale.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE 14

2.3.7. CyBlocPackSale.sol - Unused SafeERC20 library, Ownable abstract contract INFORMATIVE 16

3. VERSION HISTORY 17

1. MANAGEMENT SUMMARY

1.1. About Cyball Smart Contracts

CyBall is a football-themed, NFT-based game with a Play-to-Earn model that allows you to test your might against players from around the world

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Cyball game. It was conducted on the source code provided by the CyBall team.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The initial review was conducted on Oct 5, 2021 and a total effort of 7 working days was dedicated to identifying and documenting security issues in the code base of the Cyball Smart Contracts.

The following files were made available in the course of the review:

FILE	SHA256 SUM
AccessControl.sol	6dedd959946a74e55de0150198c69b9910fbea69
AccessControlBase.sol	e6c1862434f7f8f5fecc665a5c395fa519e92b5c
CyBlocBase.sol	2d861263fbd7def941d164c4df19b59295810f18
CyBlocCore.sol	87bf80c843548ba92918594d9b154863b2568b2a
CyBlocMentorManager.sol	29e4b4c80edc049f04c7ad782dce82c0fafa52f4
CyBlocPack.sol	f4906a1ff1e188e2d7b0ed1d628088ca1601cbc4
CyBlocPackSale.sol	05fcbfc14f58bbe0b6b551bb553d629f2ce40654
CyblocUtil.sol	02e04c999155ddd16191e7f5355fc6cf062ae6a3

2.2. Findings

2.2.1. CyBlocPackSale.sol - Reuse signature in buy function LOW

In the contract, buy function uses verifyProof function to verify max value but it doesn't check if the signature was used or not. So the caller can use an old signature with old max values to bypass required statement.

```
76  function buy(uint256 quantity, uint256 max, Proof memory _proof) ext…
    ernal payable started {
77          require(quantity > 0, "CyBlocPackSale: Invalid quantity");
78          require(verifyProof(abi.encodePacked(msg.sender, max), _proo…
    f), "CyBlocPackSale: Wrong proof");
79          require(quantity <= TOTAL_PACK.sub(numberOfSoldPack), "CyBlo…
    cPackSale: Not enough packs for you");
```

```
80          require(userPackCount[msg.sender] + quantity <= max, "CyBloc…
    PackSale: You buy too much");
81          require(userPackCount[msg.sender] + quantity <= MAX_PER_USER…
    , "CyBlocPackSale: You buy too much2");
82
83          uint totalPrice = PACK_PRICE * quantity;
84
85          require(msg.value >= totalPrice, "CyBlocPackSale: Invalid ms…
    g.value");
86
87          uint packId = numberOfSoldPack;
88
89          numberOfSoldPack += quantity;
90          userPackCount[msg.sender] += quantity;
91
92          for (uint256 i = 0; i < quantity; i++) {
93              uint256 tokenId = CYBLOC_PACK.mint(msg.sender, PACK_TYPE…
    );
94              tokenIdToPackId[tokenId] = packId + 1; // if tokenId map…
     to packId, it must be greater than 0
95              packId++;
96          }
97
98          payable(owner()).transfer(totalPrice);
99
100         if (msg.value > totalPrice) {
101             payable(msg.sender).transfer(msg.value.sub(totalPrice));
102         }
103
104         emit CyblocPackPurchased(msg.sender, quantity);
105     }
```

Snippet 1. CyBlocPackSale.sol Reuse signature in `buy` function

RECOMMENDATION

We recommend adding a module to verifyProof function which ensures signature is used once.

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

2.2.2. CyBlocPack.sol - Unsafe packs opening function LOW

The open function is used to open packs. Its parameter includes tokenId and gene that are sent from users, these parameters can be discarded and request new ones. Therefore they have enough information to not to call the method and reopen to ensure better packs.

```
78  function open(PackOpen[] memory openPacks, Proof[] memory _proofs) ex…
    ternal {
79          require(openPacks.length > 0, "CyBlocPack: empty openPacks");…

80
81          for (uint256 i = 0; i < openPacks.length; i++) {
82              PackOpen memory pack = openPacks[i];
83
84              require(ownerOf(pack.tokenId) == msg.sender, "CyBlocPack:…
     Wrong pack");
85              require(locks[pack.tokenId], "CyBlocPack: Pack must be lo…
    ck before open");
86              require(verifyProof(abi.encodePacked(pack.tokenId, pack.g…
    ene), _proofs[i]), "CyBlocPack: Wrong proof");
87
88              _burn(pack.tokenId);
89
90              NFTContract.newCyBloc(msg.sender, pack.gene, 0, 0);
91              emit CyblocPacksOpened(msg.sender, pack);                …

92          }
93      }
```

RECOMMENDATION

We recommend the server implement a mechanism to freeze the data pack after the first open function call like caching or storing them.

UPDATES

- *2021-10-31*: This issue has been acknowledged by the CyBall team and the server has been implemented a mechanism to prevent this issue.

2.3. Additional notes and recommendations

2.3.1. CyballMentorManager.sol - Useless code in finishMentor function INFORMATIVE

Require statement conditions in this function are identical to the ones in the wrapped function. It is a waste of gas.

```
57  function finishMentor(uint256 tokenId, uint256 randomNumber) external…
     {
58          require(tokenId != 0, "Oops");
59          require(CyBloc.ownerOf(tokenId) == msg.sender, "You are not o…
    wner of token");
60
61          require(uint256(keccak256(abi.encodePacked(randomNumber))) ==…
     commitments[tokenId], "Wrong random number");
62
63          CyBloc.finishMentor(msg.sender, tokenId, randomNumber);
64      }
```

Snippet 2. CyballMentorManager.sol Useless code in `finishMentor` function

In the above snippet, first two require statements are not necessary because they will be checked again inside Cyblock.finishMentor() method, implemented in CyblockCore.sol, quoted in the following snippet:

```
 91  function finishMentor(address _owner, uint256 _tokenId, uint256 _see…
     d) external whenNotPaused onlyMentorManager {
 92          require(_tokenId != 0, "Oops"); //duplicated with this state…
    ment
 93          require(ownerOf(_tokenId) == _owner, "Not your token"); //du…
    plicated with this statement
 94          require(isReadyGraduation(_tokenId), "Cannot open");
 95
 96          CyBloc storage _cb = cyblocs[_tokenId];
 97
 98          uint geneMentor1 = cyblocs[_cb.mentoredById].gene;
 99          uint geneMentor2 = cyblocs[_cb.mentoredById2].gene;
100
101          _cb.cooldownEndBlock = block.number + MENTOR_COOLDOWN_BLOCK;
102          _cb.gene = geneScientist.mixGenes(
103              geneMentor1,
104              _cb.mentoredById2 == 0 ? 0 : geneMentor2,
105              _seed
106          );
```

```
107
108          emit FinishMentor(_tokenId, _owner);
109      }
```

Snippet 3. CyblockCore.sol The wrapped function includes duplicated statements

RECOMMENDATION

Removing the first two require statements.

UPDATES

- *2021-10-31*: This issue has been acknowledged and fixed by the CyBall team.

2.3.2. CyballMentorManager.sol - Useless code in startMentor function INFORMATIVE

Require statement conditions in this function are identical to the ones in the wrapped function. It is a waste of gas.

```
57  function startMentor(uint256 _mentor, uint256 _mentor2, uint256 _comm…
    itment, Proof memory _proof) external returns (uint256) {
58          require(_mentor != 0, "Mentor is 0");
59          require(CyBloc.ownerOf(_mentor) == msg.sender, "You are not o…
    wner of mentor");
60          require(_mentor2 == 0 || CyBloc.ownerOf(_mentor2) == msg.send…
    er, "You are not owner of mentor2");
61
62          bytes memory encode = abi.encodePacked(
63              _mentor, CyBloc.getMentorCount(_mentor),
64              _mentor2, CyBloc.getMentorCount(_mentor2),
65              _commitment,
66              msg.sender
67          );
68
69          require(verifyProof(encode, _proof), "Wrong proof");
70
71          payMentorFee();
72          uint256 newCyblocId = CyBloc.startMentor(msg.sender, _mentor,…
     _mentor2);
73
74          commitments[newCyblocId] = _commitment;
75
76          return newCyblocId;
77      }
```

Snippet 4. CyballMentorManager.sol Useless code in `startMentor` function

In the above snippet, first 3 require statements are not necessary because they will be checked again inside Cyblock.startMentor() method, implemented in CyblockCore.sol, quoted in the following snippet:

```
91  function startMentor(address _owner, uint256 _mentor, uint256 _mento…
    r2) external whenNotPaused onlyMentorManager returns (uint256) …
    {
92          require(_mentor != 0, "Mentor is 0"); //duplicated with this…
     statement
93          require(ownerOf(_mentor) == _owner, "You are not owner of me…
    ntor"); //duplicated with this statement
94          require(isReadyMentor(_mentor), "Mentor is not ready");
95
96          cyblocs[_mentor].cooldownEndBlock = block.number + MENTOR_CO…
    OLDOWN_BLOCK;
97          cyblocs[_mentor].mentorCount++;
98
99          if (_mentor2 != 0) { //duplicated with this logic
100             require(_mentor2 != _mentor, "Use same mentor");
101             require(ownerOf(_mentor2) == _owner, "You are not owner …
    of mentor2");
102             require(isReadyMentor(_mentor2), "Mentor2 is not ready");
103
104             cyblocs[_mentor2].cooldownEndBlock = block.number + MENT…
    OR_COOLDOWN_BLOCK;
105             cyblocs[_mentor2].mentorCount++;
106         }
107
108         uint256 newCyblocId = _spawn(_owner, 0, _mentor, _mentor2);
109         cyblocs[newCyblocId].cooldownEndBlock = block.number + OPEN_…
    COOLDOWN_BLOCK;
110         emit StartMentor(newCyblocId, _owner);
111
112         return newCyblocId;
113     }
```

Snippet 5. CyblockCore.sol The wrapped function includes duplicated statements

RECOMMENDATION

Remove the first three require statements.

UPDATES

- *2021-10-31*: This issue has been acknowledged and fixed by the CyBall team.

2.3.3. CyBlocPack.sol - Unused SafeMath, SafeERC20, Ownable INFORMATIVE

In the head of the source code, the contract imported SafeMath, SafeErc20 libraries and Ownable abstract contract but it doesn't use inside the CyBlocPack contract.

RECOMMENDATION

We suggest removing them for readability.

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

2.3.4. CyBlocPack.sol - Inaccurate require statement message INFORMATIVE

Error message on require statement is inaccurate.

```
37  modifier onlySellerOrOwner(uint256 _packType) {
38          require(Sellers[_packType] == msg.sender || owner() == msg.se…
    nder, "CyBlockPack: wrong seller");
39          _;
40      }
```

Snippet 6. CyBlocPack.sol Inaccurate require statement message

RECOMMENDATION

Change message to wrong seller or owner.

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

2.3.5. CyBlocPack.sol - Reuse signature INFORMATIVE

The contract uses verifyProof function to verify transactions but it doesn't check if the signature was used or not. So the caller can use an old signature to make another transaction. Currently, it doesn't affect the contract but it may cause issues in future development.

RECOMMENDATION

We recommend adding a module to verifyProof function which ensures signature is used once.

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

2.3.6. CyBlocPackSale.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All safe math usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default, the only usage of safemath now is to have a custom revert message which isn't the case in the auditing contracts. We suggest to use normal operators for readability and gas saving.

In the contract, it is still used in the below function.

```
76  function buy(uint256 quantity, uint256 max, Proof memory _proof) ext…
    ernal payable started {
77          require(quantity > 0, "CyBlocPackSale: Invalid quantity");
78          require(verifyProof(abi.encodePacked(msg.sender, max), _proo…
    f), "CyBlocPackSale: Wrong proof");
79          require(quantity <= TOTAL_PACK.sub(numberOfSoldPack), "CyBlo…
    cPackSale: Not enough packs for you");
80          require(userPackCount[msg.sender] + quantity <= max, "CyBloc…
    PackSale: You buy too much");
81          require(userPackCount[msg.sender] + quantity <= MAX_PER_USER…
    , "CyBlocPackSale: You buy too much2");
82
83          uint totalPrice = PACK_PRICE * quantity;
84
85          require(msg.value >= totalPrice, "CyBlocPackSale: Invalid ms…
    g.value");
86
87          uint packId = numberOfSoldPack;
88
89          numberOfSoldPack += quantity;
90          userPackCount[msg.sender] += quantity;
91
92          for (uint256 i = 0; i < quantity; i++) {
93              uint256 tokenId = CYBLOC_PACK.mint(msg.sender, PACK_TYPE…
    );
94              tokenIdToPackId[tokenId] = packId + 1; // if tokenId map…
```

```
     to packId, it must be greater than 0
 95              packId++;
 96          }
 97
 98          payable(owner()).transfer(totalPrice);
 99
100          if (msg.value > totalPrice) {
101              payable(msg.sender).transfer(msg.value.sub(totalPrice));
102          }
103
104          emit CyblocPackPurchased(msg.sender, quantity);
105      }
```

Snippet 7. CyBlocPackSale.sol buy function recommend fixing

RECOMMENDATION

Change sub function to subtract operator.

```
76  function buy(uint256 quantity, uint256 max, Proof memory _proof) ext…
    ernal payable started {
77          require(quantity > 0, "CyBlocPackSale: Invalid quantity");
78          require(verifyProof(abi.encodePacked(msg.sender, max), _proo…
    f), "CyBlocPackSale: Wrong proof");
79          require(quantity <= TOTAL_PACK - numberOfSoldPack, "CyBlocPa…
    ckSale: Not enough packs for you");
80          require(userPackCount[msg.sender] + quantity <= max, "CyBloc…
    PackSale: You buy too much");
81          require(userPackCount[msg.sender] + quantity <= MAX_PER_USER…
    , "CyBlocPackSale: You buy too much2");
82
83          uint totalPrice = PACK_PRICE * quantity;
84
85          require(msg.value >= totalPrice, "CyBlocPackSale: Invalid ms…
    g.value");
86
87          uint packId = numberOfSoldPack;
88
89          numberOfSoldPack += quantity;
90          userPackCount[msg.sender] += quantity;
91
92          for (uint256 i = 0; i < quantity; i++) {
93              uint256 tokenId = CYBLOC_PACK.mint(msg.sender, PACK_TYPE…
```

```
    );
94              tokenIdToPackId[tokenId] = packId + 1; // if tokenId map…
     to packId, it must be greater than 0
95              packId++;
96          }
97
98          payable(owner()).transfer(totalPrice);
99
100         if (msg.value > totalPrice) {
101             payable(msg.sender).transfer(msg.value - totalPrice);
102         }
103
104         emit CyblocPackPurchased(msg.sender, quantity);
```

Snippet 8. CyBlocPackSale.sol buy function recommend fixing

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

2.3.7. CyBlocPackSale.sol - Unused SafeERC20 library, Ownable abstract contract INFORMATIVE

In the head of the source code, the contract imported SafeErc20 library and Ownable abstract contract. The SafeER20 is used for IERC20 at line 17 but the IERC20 doesn't use anywhere. Ownable abstract contract is the same.

```
16  using SafeMath for uint256;
17      using SafeERC20 for IERC20;
18
19      CyBlocPack public CYBLOC_PACK;
20      uint256 public PACK_TYPE;
```

Snippet 9. CyBlocPackSale.sol Unnecessary statement in the contract

RECOMMENDATION

We suggest removing them for readability (including the codes at line 17 which uses SaferERC20 for IERC20)

UPDATES

- *2021-11-02*: This issue has been acknowledged by the CyBall team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*2021-10-05*	Private Report	Verichains Lab
2.0	*2021-10-31*	Private Report	Verichains Lab
2.1	*2021-11-02*	Private Report	Verichains Lab
2.2	*2021-11-05*	Public Report	Verichains Lab

Table 2. Report versions history