HAECHI AUDIT

GameTokenStaking

Smart Contract Security Analysis

Published on : Aug 12, 2022

Version v2.0

HAECHI AUDIT

Smart Contract Audit Certificate

GameTokenStaking

Security Report Published by HAECHI AUDIT
v1.0 Aug 12, 2022
v2.0 Aug 22, 2022

Auditor : Taek Lee

Executive Summary

Severity of Issues	Findings	Resolved	Unresolved	Acknowledged	Comment
Critical	-	-	-	-	-
Major	2	2	-	-	-
Minor	1	1	-	-	-
Tips	7	5	-	2	-

TABLE OF CONTENTS

10 Issues (0 Critical, 2 Major, 1 Minor, 7 Tips) Found

ABOUT US

HAECHI AUDIT은 디지털 자산이 가져올 금융 혁신을 믿습니다. 디지털 자산을 쉽고 안전하게 만들기 위해 HAECHI AUDIT은 '보안'과 '신뢰'라는 가치를 제공합니다. 그로써 모든 사람이 디지털 자산을 부담없이 활용할 수 있는 세상을 꿈꿉니다.

HAECHI AUDIT은 글로벌 블록체인 업계를 선도하는 HAECHI LABS의 대표 서비스 중 하나로, 스마트 컨트랙트 보안 감사 및 개발을 전문적으로 제공합니다.

다년간 블록체인 기술 연구 개발 경험을 보유하고 있는 전문가들로 구성되어 있으며, 그 전문성을 인정받아 블록체인 기술 기업으로는 유일하게 삼성전자 스타트업 육성 프로그램에 선정된 바 있습니다. 또한, 이더리움 재단과 이더리움 커뮤니티 펀드로부터 기술 장려금을 수여받기도 하였습니다.

대표적인 클라이언트 및 파트너사로는 카카오 자회사인 Ground X, LG, 한화, 신한은행 등이 있으며, Sushiswap,1inch, Klaytn, Badger와 같은 글로벌 블록체인 프로젝트와도 협업한 바 있습니다. 지금까지 약 400여곳 이상의 클라이언트를 대상으로 가장 신뢰할 수 있는 스마트 컨트랙트 보안감사 및 개발 서비스를 제공하였습니다.

문의 : audit@haechi.io
웹사이트 : audit.haechi.io

INTRODUCTION

본 보고서는 MarbleX 팀이 제작한 GameTokenStaking 스마트 컨트랙트의 보안을 감사하기 위해 작성되었습니다. HAECHI AUDIT 는 MarbleX 팀이 제작한 스마트 컨트랙트의 구현 및 설계가 공개된 자료에 명시한 것처럼 잘 구현이 되어있고, 보안상 안전한지에 중점을 맞춰 감사를 진행했습니다.

CRITICAL	Critical 이슈는 광범위한 사용자가 피해를 볼 수 있는 치명적인 보안 결점으로 반드시 해결해야 하는 사항입니다.
MAJOR	Major 이슈는 보안상에 문제가 있거나 의도와 다른 구현으로 수정이 필요한 사항입니다.
MINOR	Minor 이슈는 잠재적으로 문제를 발생시킬 수 있으므로 수정이 요구되는 사항입니다.
TIPS	Tips 이슈는 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항입니다.

HAECHI AUDIT는 MarbleX 팀이 발견된 모든 이슈에 대하여 개선하는 것을 권장합니다.
이어지는 이슈 설명에서는 코드를 세부적으로 지칭하기 위해서 {파일 이름}#{줄 번호}, {컨트랙트 이름}#{함수/변수 이름} 포맷을 사용합니다. 예를 들면, *Sample.sol:20*은 Sample.sol 파일의 20번째 줄을 지칭하며, *Sample#fallback()* 는 Sample 컨트랙트의 fallback() 함수를 가리킵니다
보고서 작성을 위해 진행된 모든 테스트 결과는 Appendix에서 확인 하실 수 있습니다.

SUMMARY

Audit에 사용된 코드는 GitHub (https://github.com/MarblexAudit/GameTokenStaking/tree/a98f22b31f1fe5b472d903307370d7ec0b9877cf)에서 찾아볼 수 있습니다. Audit에 사용된 코드의 마지막 커밋은 “a98f22b31f1fe5b472d903307370d7ec0b9877cf”입니다.

Issues	HAECHI AUDIT에서는 Critical 이슈 0개, Major 이슈 2개, Minor 이슈 1개를 발견하였으며 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항들을 7개의 Tips 카테고리로 나누어 서술하였습니다.
Update	[v.2.0] 새로운 커밋 821a2dcc09348d6f1e48cd029ffe13c350925f07 에서 Major 이슈 2개, Minor 이슈 1개, Tips 5개가 Resolved 되었으며 2개의 Tips가 Acknowledged 상태로 변경되었습니다

Severity	Issue	Status
⚠ MAJOR	BankPair{Child/Parent}.deposit() 함수의 잘못된 구현으로 인해, Bank 기능이 마비될 수 있습니다	(Resolved - v2.0)
⚠ MAJOR	BankPairChildFactory에서 create/createWithMasterChef() 함수 실행시에, ChildBridge의 Owner를 변경하지 않습니다	(Resolved - v2.0)
◘ MINOR	BankFactory.pairs() 함수가 잘못구현되어 함수 호출에 실패합니다.	(Resolved - v2.0)
💡 TIPS	BankPairChildFactory에서 create() 함수로 BankPairChild를 생성한경우, 정상적인 작동을 보장하지 않습니다	(Resolved - v2.0)
💡 TIPS	MasterChef.claim() 함수의 updatePool() 함수의 호출 순서를 조정하십시오	(Resolved - v2.0)
💡 TIPS	BankPair{Child/Parent}Factory에서 CounterPart 브릿지를 설정하지 않습니다	(Resolved - v2.0)
💡 TIPS	BankPairParentFactory에서 initialize함수 호출에 실패 할 수 있습니다	(Acknowledged - v2.0)
💡 TIPS	MasterChef.setSeason() 함수에 _rewardToken값을 삭제하십시오	(Resolved - v2.0)
💡 TIPS	BankBridge.handleERC20Transfer()이 지원하지 않는 토큰에 대해 호출가능합니다	(Acknowledged - v2.0)
💡 TIPS	BankTransfer.bridgeEvent() 함수에 중복된 require문이 존재합니다	(Resolved - v2.0)

OVERVIEW

Contracts subject to audit

- BankPairChildFactory.sol
- BankPairParentFactory.sol
- BankBridge.sol
- BankFactory.sol
- BankPairChild.sol
- BankPairData.sol
- BankPairParent.sol
- BankToken.sol
- BankTransfer.sol
- BridgeFactory.sol
- IBankBridge.sol
- IBankPairChild.sol
- IBankPairData.sol
- IBridgeEvent.sol
- IBridgeFactory.sol
- Bridge.sol
- BridgeCounterPart.sol
- BridgeFee.sol
- BridgeHandledRequests.sol
- BridgeOperator.sol
- BridgeTokens.sol
- BridgeTransfer.sol
- BridgeTransferERC20.sol
- masterchef
- IMasterChefFactory.sol
- MasterChef.sol
- MasterChefFactory.sol
- Admin.sol
- BlockRecord.sol

FINDINGS

⚠ MAJOR

BankPair{Child/Parent}.deposit() 함수의 잘못된 구현으로 인해, Bank 기능이 마비될 수 있습니다

(Found - v.1.0)(Resolved - v.2.0)

```
    function deposit(uint256 season, uint256 amount) external onlyAdmin returns (bool) {
        // ecoWallet address is required
        require(ecoWallet != address(0), "D01");
        // childBank address is required
        require(childBank != address(0), "D02");
        // payToken address is required
        require(payToken != address(0), "D03");
        // season must be greater than 0.
        require(season > 0, "D04");
        // amount must be greater than 0.
        require(amount > 0, "D05");

        uint256 amount50p = amount / 2;

        uint256 bankBalance = IERC20(payToken).balanceOf(address(this));
        // half of the total reward amount of tokens must be present.
        require(bankBalance >= amount50p, "D06");

        IERC20(payToken).transferFrom(ecoWallet, address(this), amount50p);

        ERC20PresetMinterPauser(bridgeToken).mint(address(this), amount);

        bytes memory extraData = encodeDeposit(season, amount);
        _sendBridge(childBank, amount, extraData);

        _addBank(season, bridgeToken, amount);

        emit Deposit(bridgeToken, season, amount);
        _increaseBlock();
        return true;
    }
```

[https://github.com/MarblexAudit/GameTokenStaking/blob/a98f22b31f1fe5b472d903307370d7ec0b9877cf/contracts/bank/BankPairParent.sol#L50]

```
    function deposit(
        address token,
        uint256 season,
        uint256 amount
    ) external onlyAdmin returns (bool) {
        // ecoWallet address is required
        require(ecoWallet != address(0), "D01");
        // token address is required
        require(token != address(0), "D02");
        // season must be greater than 0.
        require(season > 0, "D03");
        // amount must be greater than 0.
```

```
        require(amount > 0, "D04");

        uint256 amount50p = amount / 2;

        uint256 bankBalance = IERC20(token).balanceOf(address(this));
        // half of the total reward amount of tokens must be present.
        require(bankBalance >= amount50p, "D05");

        IERC20(token).transferFrom(ecoWallet, address(this), amount50p);

        _addBank(season, token, amount);

        emit Deposit(token, season, amount);
        _increaseBlock();
        return true;
    }
```

[https://github.com/MarblexAudit/GameTokenStaking/blob/a98f22b31f1fe5b472d903307370d7ec0b9877cf/contracts/bank/BankPairChild.sol#L60]

Issue

BankPairChild 와 BankPairParent의 deposit함수는, Admin이 MasterChef에 보상을 deposit 하는데 사용됩니다.

이 과정에서, ecoWallet에서 절반을 가져오고 나머지 절반을 Bank의 잔액을 확인하여 deposit 하게됩니다.

하지만, Bank의 잔액을 계산하는 과정이 모든 deposit 과정에서 중복되어 사용되다보니, 실제 Admin이 예치한 토큰 양보다 많이 예치하였다고 계산됩니다.

따라서 실제 예치된 토큰보다 많이 출금 요청이 있는경우 bridge로 작동할 수 없게 됩니다

Recommendation

잔액을 확인하는 방식이 아닌 msg.sender로 부터 토큰을 받아오는 방식을 활용하여 deposit기능을 수행하십시오.

Update

[v.2.0] - MarbleX 팀에서 deposit 함수로 누적된 금액을 저장하는 방식으로 해당 이슈를 해결하였습니다

⚠ MAJOR

BankPairChildFactory에서 create/createWithMasterChef() 함수 실행시에, ChildBridge의 Owner를 변경하지 않습니다

(Found - v.1.0)(Resolved - v.2.0)

```
    function createWithMasterChef(
        address childOperator,
        address parentBridge,
        address parentToken,
        address parentBank,
        address newBankOwner,
        address newEcoWallet,
        address stakeToken,
        address feeWallet,
        string memory tokenName,
        string memory tokenSymbol
    ) external onlyOwner returns (address, address) {
        require(masterChefFactory != address(0), "CWMC01");

        address newBank = _create(
            childOperator,
            parentBridge,
            parentToken,
            parentBank,
            newBankOwner,
            newEcoWallet,
            tokenName,
            tokenSymbol
        );
        address newMasterChef = IMasterChefFactory(masterChefFactory).create(newBank, stakeToken,
feeWallet);
        BankPairChild(newBank).setMasterChef(newMasterChef);
        Ownable(newBank).transferOwnership(newBankOwner);

        return (newBank, newMasterChef);
    }
```

[https://github.com/MarblexAudit/GameTokenStaking/blob/a98f22b31f1fe5b472d903307370d7ec0b9877cf/contracts/BankPairChildFactory.sol#L20]

```
    function create(
        address childOperator,
        address parentBridge,
        address parentToken,
        address parentBank,
        address newBankOwner,
        address newEcoWallet,
        string memory tokenName,
        string memory tokenSymbol
    ) public onlyOwner returns (address) {
        address newBank = _create(
            childOperator,
            parentBridge,
            parentToken,
            parentBank,
            newBankOwner,
            newEcoWallet,
            tokenName,
```

```
            tokenSymbol
        );

        Ownable(newBank).transferOwnership(newBankOwner);

        return newBank;
    }
```

[https://github.com/MarblexAudit/GameTokenStaking/blob/a98f22b31f1fe5b472d903307370d7ec0b9877cf/contracts/BankPairChildFactory.sol#L51]

Issue

BankPairChildFactory.create/createWithMasterChef() 함수는 Game Chain에서 bridge, bank, masterChef를 배포하고 초기화 하는 함수입니다.

이때 배포하게 되는 컨트랙트들의 소유권을 Factory에서 소유하다가 지정한 다른 주소(newBankOwner)로 옮기게 됩니다.

이 과정에서 bank, masterChef의 소유권은 정상적으로 옮겨지지만, bridge의 소유권을 옮기는 함수는 누락되어있습니다.

bridge의 소유권을 BankPairFactory에서 다른 주소로 옮기는 함수가 존재하지 않으므로, bridge에 대한 운영관련 함수들을 더이상 사용할 수 없게 됩니다.

Recommendation

create/createWithMasterChef() 함수에 bridge의 소유권을 옮기는 실행을 추가하십시오

Update

[v.2.0] - create 함수는 삭제되고, createWithMasterChef 함수에서 bridge의 소유권을 옮기는 수행이 추가되어 이슈가 해결되었습니다

MINOR

BankFactory.pairs() 함수가 잘못구현되어 함수 호출에 실패합니다.

(Found - v.1.0)(Resolved - v.2.0)

```
    function pairs(uint256 startIdx, uint256 endIdx) external view returns (Pair[]
memory) {
        Pair[] memory ret = new Pair[](endIdx - startIdx + 1);

        for (uint256 i = startIdx; i <= endIdx; i++) {
            ret[i] = pair[i];
        }
        return ret;
    }
```

[https://github.com/MarblexAudit/GameTokenStaking/blob/a98f22b31f1fe5b472d903307370d7ec0b9877cf/contracts/bank/BankFactory.sol#L26]

Issue

BankFactory.pairs() 함수는 현재 셋팅된 pairs 의 일부를 반환하는 view 함수입니다.
조회시 startIdx, endIdx를 넣어 조회하고자 하는 범위를 입력합니다.
이 과정에서, 새로운 배열(ret)을 생성하여 값을 반환하는데, ret의 경우 startIdx가 아닌 0 부터 시작하고, 총 길이가 endIdx- startIdx +1 이 됩니다. 하지만 for 반복문안에서 ret배열에 대해 startIdx부터 값을 셋팅하도록 구현되어있어, 많은 경우 pairs() 함수가 실행에 실패합니다
만약 startIdx = 3, endIdx = 4 인 경우 ret이 길이 2인 배열이지만 반복문안에서 ret[3]에 접근하게 되어 Out-of-Index 에러를 발생시킵니다

Recommendation

ret 값을 세팅하는 구문을 ret[i-startIdx] = pair[i] 로 변경하십시오

Update

[v.2.0] - MarbleX팀에서 Recommendation을 반영하여 이슈가 해결되었습니다

💡 **TIPS**

BankPairChildFactory에서 create() 함수로 BankPairChild를 생성한경우, 정상적인 작동을 보장하지 않습니다

(Found - v.1.0)(Resolved - v.2.0)

Issue

BankPairChildFactory.create() 함수는 MasterChef 없이 배포하는 함수입니다. MasterChef가 없이 BankPairChild를 생성하게 되면 MasterChef만 호출 할 수 있는 함수들을 호출 할 수 없습니다. 추후 별도로 MasterChef를 배포하고 setMasterChef() 함수를 통해 설정할 수 있지만, 별도로 배포된 MasterChef에 대해서는 정상적인 작동을 보장 하지 않습니다.

Recommendation

BankPairChildFactory.create()함수를 삭제하는 것을 추천드립니다

Update

[v.2.0] - MarbleX팀에서 Recommendation을 반영하여 이슈가 해결되었습니다

💡 **TIPS**

MasterChef.claim() 함수의 updatePool() 함수의 호출 순서를 조정하십시오

(Found - v.1.0)(Resolved - v.2.0)

Issue

MasterChef.claim() 함수는, 시즌 중에 보상 토큰을 수령하는 함수입니다. 이 과정에서 예치금에 비례하는 리워드 금액을 뜻하는 accTokenPerShare값이 0인지 확인하는 방식으로 시즌이 시작됐는지 확인합니다. 하지만 모두가 preSeason에 deposit을 하고 시즌중에 아무도 deposit/withdraw를 안했지만 시즌중인경우 아무도 updatePool함수를 호출하지 않은 상황이라 accTokenPerShare값이 0으로 고정되어있습니다. 따라서 정상적으로 claim받을 수 있어야 하는 상황임에도 불구하고 리워드를 받지 못하게 됩니다.

Recommendation

claim() 함수 호출 후 바로 updatePool() 함수를 실행하도록 실행순서를 변경하십시오

Update

[v.2.0] - MarbleX팀에서 Recommendation을 반영하여 이슈가 해결되었습니다

💡 **TIPS**

BankPair{Child/Parent}Factory에서 CounterPart 브릿지를 설정하지 않습니다

(Found - v.1.0)(Resolved - v.2.0)

Issue

BankPair{Child/Parent}Factory에서 bridge에 연결된 주소들을 array로 관리하고 있습니다. 이 과정에서 parentBridge와 childBridge도 index에 연결하는 방식으로 관리합니다.
다만, BridgeCounterPart라는 컨트랙트를 이미 상속받는 상황에서 이를 활용하여 값을 관리하는 것이 명시적일 것으로 파악됩니다.

Recommendation

BankPair{Child/Parent}Factory에서 setCounterPartBridge() 함수로 CounterPart를 설정하시는 것을 추천드립니다.

Update

[v.2.0] - MarbleX팀에서 Recommendation을 반영하여 이슈가 해결되었습니다

💡**TIPS**

BankPairParentFactory에서 initialize함수 호출에 실패 할 수 있습니다

(Found - v.1.0)(Acknowledged - v.2.0)

Issue

BankPairParentFactory에서는 create() 함수로 bank/bridge컨트랙트를 배포하고 initialize() 함수로 child와 연결하도록 되어있습니다.
만약 initialize함수를 호출하기 이전에 operator가 본인의 operator권한을 revoke 한다면 operatorList 배열의 길이가 1이 되어 1번 index에 접근할 수 없게 되어 Out-of-Index 에러를 발생시키고 실행이 취소됩니다

Recommendation

Bridge의 Owner의 주소를 initialize 함수 호출시에 입력받는것으로 변경하십시오

Update

[v.2.0] - MarbleX팀에서 해당이슈를 인지하고 있지만, 해당 함수 호출자의 역량으로 판단하여 코드 변경을 진행하지는 않았습니다.

💡 **TIPS**

MasterChef.setSeason() 함수에 _rewardToken값을 삭제하십시오

(Found - v.1.0)(Resolved - v.2.0)

Issue

MasterChef.setSeason()은, MasterChef에서 새로운 Season값을 설정하는 용도로 사용됩니다. 이때, _rewardToken값을 입력으로 받아 해당 Season의 보상 토큰을 설정하게 되는데, 이 값은 BankPairChild에 설정된 Season 보상토큰과 일치해야하지만 확인하지 않습니다.

Recommendation

해당 값이 이미 Bank에 저장되어있으므로, 일관적인 동작을 위해 SeasonInfo 에서 rewardToken을 삭제하고 setSeason에서도 입력받지 않도록 변경하십시오.

Update

[v.2.0] - MarbleX팀에서 setSeaon 함수 호출시 _rewardToken값이 BankPairChild에 설정된 Season 보상토큰과 일치하는지 확인하여 이슈가 해결되었습니다.

💡 **TIPS**

BankBridge.handleERC20Transfer()이 지원하지 않는 토큰에 대해 호출가능합니다

(Found - v.1.0)(Acknowledged - v.2.0)

Issue

BankBridge.handleERC20Transfer() 함수는 BridgeTransferERC20.handleERC20Transfer() 함수를 사용합니다. 하지만 해당 함수는 주어진 토큰이 등록되었는지 확인하지 않으며 lock 되어있는지 확인하지도 않습니다.

Recommendation

BankBridge.handleERC20Transfer() 함수실행시 주어진 토큰이 Bridge에 등록된 토큰인지 lock되어있는지 확인하시는 것을 추천드립니다

Update

[v.2.0] - MarbleX팀에서 해당이슈를 인지하고 있지만, 브릿지에서 해당 이벤트를 구독하지 않는다고 하여 코드 변경을 진행하지는 않았습니다.

💡**TIPS**

BankTransfer.bridgeEvent() 함수에 중복된 require문이 존재합니다

(Found - v.1.0)(Resolved - v.2.0)

Issue

BankTransfer.bridgeEvent() 함수에 현재 bridge주소가 설정되어있는지, 호출자가 bridge인지 확인하는 구문이 들어있습니다. 하지만 onlyBridge modifier에서 이미 해당 값들을 확인하고 있으며 bridgeEvent 함수에서 onlyBridge modifier를 사용하고 있습니다.

Recommendation

중복된 require문을 삭제하십시오

Update

[v.2.0] - MarbleX팀에서 Recommendation을 반영하여 이슈가 해결되었습니다

DISCLAIMER

해당 리포트는 투자에 대한 조언, 비즈니스 모델의 적합성, 버그 없이 안전한 코드를 보증하지 않습니다. 해당 리포트는 알려진 기술 문제들에 대한 논의의 목적으로만 사용됩니다. 리포트에 기술된 문제 외에도 메인넷 상의 결함 등 발견되지 않은 문제들이 있을 수 있습니다. 안전한 스마트 컨트랙트를 작성하기 위해서는 발견된 문제들에 대한 수정과 충분한 테스트가 필요합니다.

End of Document