SALUS SECURITY

DEC 2023

CODE SECURITY ASSESSMENT

X WINNER

Overview

Project Summary

- Name: X Winner - powerball and pokerBaccarat
- Platform: Arbitrum
- Language: Solidity
- Repository:
    - https://github.com/xwinner-dao/platform-contracts
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	X Winner - powerball and pokerBaccarat
Version	v3
Type	Solidity
Dates	Dec 21 2023
Logs	Dec 15 2023; Dec 19 2023; Dec 21 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	3
Total informational issues	3
Total	7

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Rewards may be paid beyond expectations	Medium	Business Logic	Resolved
2	Incorrect conditional statement	Low	Business Logic	Resolved
3	A mismatch between implementation and documentation	Low	Inconsistency	Resolved
4	Centralization risk	Low	Centralization	Acknowledged
5	Powerball prize distribution may be unfair under certain circumstances	Informational	Business Logic	Acknowledged
6	Typo	Informational	Code Quality	Resolved
7	Redundant code	Informational	Redundancy	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Rewards may be paid beyond expectations

Severity: Medium	Category: Business Logic
Target: - contracts/powerball/Powerball.sol	

Description

If the current lottery funds are insufficient to distribute rewards to all winners, the reward amounts for different levels are recalculated. In this case, higher-level users are unable to receive their corresponding rewards and can only claim rewards at a lower level.

contracts/powerball/Powerball.sol:L165-L173

```
// Check user is claiming the correct bracket
require(rewardForTicketId != 0, "No prize for this bracket");

if (_brackets[i] != 5) {
        require(
        _calculateRewardsForTicketId(_lotteryId, thisTicketId, _brackets[i] + 1) == 0,
        "Bracket must be higher"
        );
}
```

contracts/powerball/Powerball.sol:L510-L536

```
function _calculateRewardsForTicketId(
        uint256 _lotteryId,
        uint256 _ticketId,
        uint32 _bracket
) internal view returns (uint256) {
        ...
        uint32 transformedUserNumber = _bracketCalculator[_bracket] + (userNumber %
(uint32(10) ** (_bracket + 1)));

        // Confirm that the two transformed numbers are the same, if not throw
        if (transformedWinningNumber == transformedUserNumber) {
            return lotteries[_lotteryId].perRewards[_bracket];
        } else {
            return 0;
        }
}
```

However, when recalculating perReward, only the number of winners at the current level is taken into account. Since higher-level rewards are set to 0, higher-level users can only claim rewards at a lower level. In this case, the number of users claiming certain rewards will be more than countWinners[i], causing the total amount claimed by all users to exceed the expected value.

contracts/powerball/Powerball.sol:L269-L284

```
function drawFinalNumberAndMakeLotteryClaimable() external override nonReentrant {
        ...
        else {
            amountToShareToWinners = 0;
            for (uint8 i = 0; i < 5; ++i) {
                amountToShareToWinners += countWinners[i] * perRewards[i];
                if (amountToShareToWinners > amountCollected) {
                    amountToShareToWinners -= countWinners[i] * perRewards[i];
                    perRewards[i] = (amountCollected - amountToShareToWinners) /
countWinners[i];
                    for (uint8 j = i + 1; j < 5; ++j) {
                        perRewards[j] = 0;
                    }
                    perRewards[5] = 0;
                    pendingInjectionNextLottery = 0;
                    break;
                }
            }
        }
        ...
}
```

Recommendation

It is advisable that when recalculating perReward, if the rewards for higher level users cannot be fulfilled, the number of these higher level users should be considered when calculating the reward of lower levels.

Status

This issue has been resolved by the team with commit 22cf064.

2. Incorrect conditional statement

Severity: Low	Category: Business Logic
Target: - contracts/powerball/Powerball.sol	

Description

In a new lotteryId, the valid ticketId should range from “firstTicketId” (inclusive within this lotteryId) to “firstTicketIdNextLottery” (inclusive within the next lotteryId).

For an invalid ticketId, it should either be less than “firstTicketId” or greater than or equal to “firstTicketIdNextLottery”. Only one of these conditions can be satisfied, making it impossible for the current logic to fulfill both criteria simultaneously. Additionally, there are issues with the boundary, causing improperly filtering out invalid ticketId values.

contracts/powerball/Powerball.sol:L435-L440

```
function viewRewardsForTicketId(
        uint256 _lotteryId,
        uint256 _ticketId,
        uint32 _bracket
    ) external view returns (uint256) {
        ...

        if (
            (lotteries[_lotteryId].firstTicketIdNextLottery < _ticketId) &&
            (lotteries[_lotteryId].firstTicketId >= _ticketId)
        ) {
            return 0;
        }

        return _calculateRewardsForTicketId(_lotteryId, _ticketId, _bracket);
}
```

Recommendation

Consider adjusting the logic, no longer requiring both conditions to be met at the same time, but using the logical operator ||, and also setting up correct boundary checks.

Can change `((lotteries[_lotteryId].firstTicketIdNextLottery < _ticketId) && (lotteries[_lotteryId].firstTicketId >= _ticketId))`
to
`((lotteries[_lotteryId].firstTicketIdNextLottery <= _ticketId) || (lotteries[_lotteryId].firstTicketId > _ticketId))`.

Status

This issue has been resolved by the team with commit 37af95b.

3. A mismatch between implementation and documentation

Severity: Low	Category: Inconsistency

Target:
- contracts/chance/pokerBaccarat/PokerBaccarat.sol

Description

In one round, the betting limit for both the player and banker is reflected as 1200 USDT in the contract, but it is stipulated as 1000 USDT in the documentation. The betting limit for a tie is displayed as 2000 USDT in the contract, whereas the documentation specifies it as 500 USDT.

contracts/chance/pokerBaccarat/PokerBaccarat.sol:L62-L72

```
uint256[9] public bettingLimit = [
    0,
    1200_000000,
    1200_000000,
    2000_000000,
    20000_000000,
    15000_000000,
    10500_000000,
    2400_000000,
    203_000000
];
```

Recommendation

Consider fixing inconsistencies between code and documentation.

Status

This issue has been resolved by the team with commit bd35d13.

4. Centralization risk

Severity: Low	Category: Centralization

Target:
- contracts/powerball/Powerball.sol
- contracts/chance/pokerBaccarat/PokerBaccarat.sol
- contracts/pools/chanceGame/PrizePool.sol

Description

While setting values, there is no restriction imposed on a specific numerical value, e.g. Powerball.setTicketPrice(), Powerball.setRewards(). Unreasonable configurations may potentially harm user interests. The owner can designate a specific address for receiving token transfers.

contracts/powerball/Powerball.sol:L372-L376

```
function setTreasury(address _treasury) external onlyOwner {
        require(_treasury != address(0), "Cannot be zero address");

        treasury = _treasury;
}
```

contracts/pools/chanceGame/PrizePool.sol:L32-L35

```
function setRecipient(address recipient_) external onlyOwner {
        require(recipient_ != address(0), "ZeroAddress");
        recipient = recipient_;
}
```

Additionally, the team added an emergencyWithdraw() function with commit b106300. This function enables the owner to withdraw all ft tokens from the contract 90 days after applyEmergency() is invoked.

If the owner's private key is compromised by attackers, they can manipulate the contract by exercising owner permissions to set unreasonable values or transfer tokens from the contract to a contract controlled by the attacker.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

5. Powerball prize distribution may be unfair under certain circumstances

Severity: Information	Category: Business Logic

Target:
- contracts/powerball/Powerball.sol

Description

In the drawFinalNumberAndMakeLotteryClaimable() function, the finalNumber of the current round of the lottery game is determined and the reward amounts of different levels are calculated based on the amount. However, due to flaws in the reward mechanism, the reward distribution may be unfair, details are as follows:

1. When the total amount of rewards excluding the highest-level reward is less than “amountCollected”, the highest-level reward “perReward[5]” may be lower than other rewards.
2. When the total amount of rewards excluding the highest-level reward is greater than or equal to “amountCollected”, the system prioritizes the calculation of lower-level rewards, and sets higher-level rewards to 0. This results in users at higher levels only receiving rewards at lower levels, contrary to the expected rewards.

contracts/powerball/Powerball.sol:L265-L284

```
function drawFinalNumberAndMakeLotteryClaimable() external override nonReentrant {
        ...
        if (amountToShareToWinners < amountCollected) {
            perRewards[5] = (amountCollected - amountToShareToWinners) / 2 /
max(countWinners[5], 1);
            amountToShareToWinners += countWinners[5] * perRewards[5];
            pendingInjectionNextLottery = amountCollected - amountToShareToWinners;
        } else {
            amountToShareToWinners = 0;
            for (uint8 i = 0; i < 5; ++i) {
                amountToShareToWinners += countWinners[i] * perRewards[i];
                if (amountToShareToWinners > amountCollected) {
                    amountToShareToWinners -= countWinners[i] * perRewards[i];
                    perRewards[i] = (amountCollected - amountToShareToWinners) /
countWinners[i];
                    for (uint8 j = i + 1; j < 5; ++j) {
                        perRewards[j] = 0;
                    }
                    perRewards[5] = 0;
                    pendingInjectionNextLottery = 0;
                    break;
                }
            }
        }
        ...
}
```

Recommendation

Here are the following suggestions:

1. In order to ensure fairness in the game, consider refactoring the reward mechanism to prioritize the calculation of higher level rewards and progressively decrease to lower levels. This ensures that users corresponding to each level receive the appropriate rewards.
2. It is advisable to set a minimum value for perReward for each level, rather than assigning it as 0. This guarantees differentiation in rewards across levels and strives to ensure that amountCollected satisfies the sum of the minimum values for these perReward values.

Status

This issue has been acknowledged by the team. The team claimed to have off-chain methods to guarantee user rights.

6. Typo

Severity: Informational	Category: Code Quality
Target: - contracts/powerball/Powerball.sol	

Description

contracts/powerball/Powerball.sol:L86

```
constructor(address _ft, address _randomGenerator, address _platfrom) Roles(msg.sender)
```

The word "_platfrom" is misspelled, and should be "_platform" instead.

Recommendation

Consider correcting the spelling mistakes.

Status

This issue has been resolved by the team with commit 83f739b.

7. Redundant code

Severity: Informational	Category: Redundancy

Target:
- contracts/pools/chanceGame/PrizePool.sol
- contracts/powerball/powerBall.sol

Description

The state variable fund only increases when depositing but never decreases. The maximum value will be reached at a certain moment and the depositFunds() function can not be used anymore.

contracts/pools/chanceGame/PrizePool.sol:L50-L55

```
function depositFunds(uint256 amount) external {
    FT.safeTransferFrom(_msgSender(), address(this), amount);
    fund += amount;
    emit EventDepositFunds(_msgSender(), amount);
}
```

The AdminTokenRecovery event is unused.

contracts/powerball/Powerball.sol:L77

```
event AdminTokenRecovery(address token, uint256 amount);
```

Recommendation

Consider removing the state variable fund and the AdminTokenRecovery event.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 2b56a8a:

File	SHA-1 hash
contracts/chance/pokerBaccarat/PokerBaccarat.sol	977d46adf159617db5bbedc79de472567f83160d
contracts/chance/pokerBaccarat/PokerCard.sol	66ec5e31c53936ba24041a23bd10c73d19b06e07
contracts/chance/pokerBaccarat/PokerRecognizer.sol	53c0d67a0432f8f70d4dbbe42102904a3d1fdba5
contracts/pools/Pool.sol	0066b5d38dc47ce547606b9733ec89b967f647ad
contracts/pools/chanceGame/PrizePool.sol	8e7a61f99b33ecea518de569a214a1d15437226c
contracts/powerball/Powerball.sol	69adbb89bb49ec11fc658f43e8c65864d6c5101e
contracts/powerball/RandomNumberGenerator.sol	5134ed9b58c041d09dbd105a148fea7a247bd2f0
contracts/shareCalculator/ShareCalculator2.sol	db3933c83e72fbb6a9332d3747fdfc138c7476a6