# X-WINNER Platform Report

Dec 30, 2023

by **Canary Technologies Inc.**

Figure 1: X-WINNER Platform

This report presents our engineering engagement with the X-WINNER Platform team on their range-bound liquidity protocol.

| Project Name | X-WINNER Platform |
|---|---|
| Repository Link | https://github.com/xwinner-dao/platform-contracts |
| Commit Hash | First: ed10ead; Final: 4f71ce2; |
| Language | Solidity |
| Chain | Polygon |

## About Canary Technologies

The mission of Canary Technologies Inc. is to secure Web3 for all. We are powered by world-class technical capabilities and acumen, which allows us to perform granular security audits for Web3 applications and protocols. We not only audit clients' codes from a technical perspective, but also consider the connection between the codes and business vision. Our team of experts is committed to helping Web3 innovators bring value to society in the most trustworthy way possible.

## Table of Contents

## Service Scope

### Service Stages

Our auditing service includes the following two stages:

- Pre-Audit Consulting Service
- Smart Contract Auditing Service

1. **Pre-Audit Consulting Service**
    - [Protocol Security & Design Discussion Meeting] As a part of the audit service, the Canary Technologies team worked closely with the X-WINNER Platform development team to discuss potential vulnerabilities and smart contract development best practices in a timely fashion. The Canary Technologies team is very appreciative of establishing an efficient and effective communication channel with the X-WINNER Platform team, as new findings were exchanged promptly and fixes were deployed quickly, during the preliminary report stage.
2. **Smart Contract Auditing Service** The Canary Technologies team analyzed the entire project using a detailed-oriented approach to capture the fundamental logic and suggested improvements to the existing code. Details can be found under **Findings & Improvement Suggestions**.

### Methodology

- Code Assessment
    - We evaluate the overall quality of the code and comments as well as the architecture of the repository.
    - We help the project dev team improve the overall quality of the repository by providing suggestions on refactorization to follow the best practice of Web3 software engineering.
- Code Logic Analysis
    - We dive into the data structures and algorithms in the repository and provide suggestions to improve the data structures and algorithms for the lower time and space complexities.
    - We analyze the hierarchy among multiple modules and the relations among the source code files in the repository and provide suggestions to improve the code architecture with better readability, reusability, and extensibility.

- Business Logic Analysis
  - We study the technical whitepaper and other documents of the project and compare its specification with the functionality implemented in the code for any potential mismatch between them.
  - We analyze the risks and potential vulnerabilities in the business logic and make suggestions to improve the robustness of the project.
- Access Control Analysis
  - We perform a comprehensive assessment of the special roles of the project, including their authority and privileges.
  - We provide suggestions regarding the best practice of privilege role management according to the standard operating procedures (SOP).
- Off-Chain Components Analysis
  - We analyze the off-chain modules that are interacting with the on-chain functionalities and provide suggestions according to the SOP.
  - We conduct a comprehensive investigation for potential risks and hacks that may happen on the off-chain components and provide suggestions for patches.

### Audit Scope

Our auditing for X-WINNER Platform covered the repository:

- https://github.com/xwinner-dao/platform-contracts: commit hash **4f71ce2**

## Project Summary

This project focuses on auditing two new Web3 games by XWINNER: Powerball and Baccarat. We identified key issues and provided recommendations.

## Findings & Improvement Suggestions

| Severity | **Total** | **Acknowledged** | **Resolved** |
|---|---|---|---|
| **High** | 1 | 0 | 0 |
| **Medium** | 1 | 1 | 0 |
| **Low** | 8 | 3 | 3 |
| **Enhancement** | 5 | 5 | 0 |

## High

1. **No failover mechanism implemented for oracle**

| Severity | **High** |
|---|---|
| Source | PokerBaccarat.sol#L275 |
| Commit | N/A |
| Status | N/A |

### Description

The current implementation relies on a single oracle source (Chainlink VRF) for generating random words. While Chainlink VRF is a reputable oracle provider, it's recommended to enhance the security and resilience of the system by considering the use of multiple oracles for randomness.

### Exploit Scenario

N/A

### Recommendations

Explore the feasibility of integrating multiple oracles for random number generation. Design and implement a failover mechanism to handle scenarios where one oracle is unavailable. Consider handling situations where one oracle may experience downtime or issues, ensuring the continued reliability of your application.

### Results

Pending;

## Medium

1. **Business Process Conflit: Set state variables while the game is running**

| Severity | **Medium** |
|---|---|
| Source | PokerBaccarat.sol#L122-L190 |
| Commit | n/a; |

| Severity | **Medium** |
|---|---|
| Status | Acknowledged;The XWinner team confirms that they will only modify state variables when there are no active users. Also, they provide a function `injectFunds()` to add funds to the price pool. |

### Description

The owner can set or change state variables while the game is running. This may affect the ongoing game and the already participating users.

### Exploit Scenario

N/A

### Recommendations

Require the game to be finished (not running nor paused) before setting state variables.

### Results

Pending;

## Low

1. **Event `AdminTokenRecovery()` declared but never used**

| Severity | **Low** |
|---|---|
| Source | Powerball.sol#L77 |
| Commit | 60ab648; |
| Status | Fixed; |

### Description

### Exploit Scenario

N/A

### Recommendations

### Results

Pending;

2. **Lack of re-entrancy guard in `PokerBaccarat::settleWinner()`**

| Severity | **Low** |
|---|---|
| Source | PokerBaccarat.sol#L28 |
| Commit | n/a; |
| Status | Acknowledged; |

### Description

The `settleWinner` function calls to an external contract `prizePool` and has state transition later. To make sure `prizePool` does not re-enter the `settleWinner` function, a re-entrancy guard is recommended.

### Exploit Scenario

N/A

### Recommendations

Consider using the OpenZeppelin `ReentrancyGuard.sol` and add the `nonReentrant` modifier.

```
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
function settleWinner(uint256 round, Option winner, uint256 index) public nonReentrant {
}
```

### Results

Pending;

3. **Update fund value in `withdrawByOwner` function**

| Severity | **Low** |
|---|---|
| Source | PrizePool.sol#L46 |
| Commit | n/a; |

| Severity | **Low** |
|---|---|
| The XWinner team confirms that the state variable is for tracking the amount of funds injected by XWinner, which does not influence the actual `ft` token stored in the contract. | |
| Status | Acknowledged; |

### Description

`fund` should update the value by `fund -= amount;`

### Exploit Scenario

N/A

### Recommendations

### Results

Pending;

4. **Magic Number: `_odd` decimal**

| Severity | **Low** |
|---|---|
| Source | PokerBaccarat.sol#L73 |
| PokerBaccarat.sol#L297 | |
| Commit | 60ab648; |
| Status | Fixed; |

### Description

Contract `PokerBaccarat.sol` uses the state variable `uint16 odds` to represent the odds of the game.

```
uint16[9] public odds = [0, 196, 196, 2300, 220, 300, 460, 2000, 24600];
```

In the actual calculation, the odd of the game is x1.96 when the state variable is `196`, that is, the decimal of the state variable `odd` is implicitly set as `100`.

```
amount = (amount * _odds) / 100;
```

**Exploit Scenario**

N/A

**Recommendations**

The decimal of the state variable `odd` should be explicitly set as `100`. Suggest introducing a constant `ODD_DECIMAL = 2`.

**Results**

Pending;

5. **Lack of event emission for game configuration changes**

| Severity | **Low** |
|---|---|
| Source | PokerBaccarat.sol#L109 |
| Commit | n/a; |
| Status | Acknowledge; |

**Description**

The PokerBaccarat contract currently emits events for transparency. We propose adding additional events for configurational changes such as when price, betting limits, and odds are changed as they directly influence potential winnings

**Exploit Scenario**

N/A

**Recommendations**

Add the following events

```
event EventBettingLimitChanged(uint256 indexed index, uint256 limit);
event EventPriceChanged(uint256 minPrice, uint256 maxPrice);
event EventOddsChanged(uint256 indexed index, uint16 odds);
```

**Results**

Pending;

6. **Lack of event emission for maximum ticket allowance changes**

| Severity | **Low** |
|---|---|
| Source | Powerball.sol#L86 |
| Commit | xxxxxxx; |
| Status | Pending; |

**Description**

Setting max number ticket is related to the business of the platform therefore, it is better to inform users by emitting an event.

**Exploit Scenario**

N/A

**Recommendations**

Add event emit to this function therefore, the user will be informed about the status of contract

**Results**

Pending;

7. **Lack of input sanitizer in Powerball contract constructor**

| Severity | **Low** |
|---|---|
| Source | Powerball.sol#L86 |
| Commit | n/a; |
| Status | Acknowledged; |

**Description**

`Powerball` contract has a constructor that accepts three variables as input. It is possible to pass a zero address accidentally to the constructor. Note that there is no setter for the `platform` address and `ft` token address.

```
constructor(address _ft, address _randomGenerator, address _platfrom) Roles(msg.sender) {
    ft = IERC20(_ft);
    randomGenerator = IRandomNumberGenerator(_randomGenerator);
    platform = _platfrom;
    treasury = msg.sender;
    }
```

**Exploit Scenario**

N/A

**Recommendations**

Check these addresses not to be equal to address zero in the constructor code.

**Results**

Pending;

8. **Mismatch `require()` check and message**

| Severity | **Low** |
|---|---|
| Source | Powerball.sol#L73 |
| Commit | 60ab648 |
| Status | Fixed; |

**Description**

```
require(msg.sender == tx.origin, "Contract not allowed");
```

cannot block transparent proxy. `delegatecall` passes `msg.sender` as `tx.origin`. You simply cannot block a transparent proxy.

**Exploit Scenario**

N/A

**Recommendations**

```
require(msg.sender == tx.origin, "Contract not allowed");
```

**Results**

Pending;

### Enhancement

1. **Optimizing gas by reducing the number of state changes**

| Severity | **Enhancement** |
|---|---|
| Source | Powerball.sol#L110-L128 |
| Commit | n/a; |
| Status | Acknowledged; |

### Description

The buyTickets function is likely called frequently and involves too many `numberTicketsPerLotteryId[_lotteryId][...]` state changes which is gas fee costing.

### Exploit Scenario

N/A

### Recommendations

For buyTickets, instead of updating `numberTicketsPerLotteryId[_lotteryId][...]` inside the loop, accumulate changes in a local variable and then apply them to the state variable after the loop. This would reduce the number of state changes, which are costly in terms of gas.

```
// Create a temporary in-memory array to store updates
uint256[1111111] memory updates;

for (uint256 i = 0; i < length; ++i) {
    uint32 thisTicketNumber = _ticketNumbers[i];

    // ... Validation checks ...

    updates[1 + (thisTicketNumber % 10)] += 1;
    updates[11 + (thisTicketNumber % 100)] += 1;
    // ... More accumulations to updates array ...

    // ... Other operations ...
}
```

```
// Apply the accumulated updates to the state variable
for (uint256 j = 1; j <= 1111111; ++j) {
    if (updates[j] > 0) {
        numberTicketsPerLotteryId[_lotteryId][j] += updates[j];
    }
}
```

**Results**

Pending;

2. **Violate single responsibility practice indrawFinalNumberAndMakeLotteryClaimable function**

| Severity | **Enhancement** |
|---|---|
| Source | Powerball.sol#L223-L301 |
| Commit | n/a; |
| Status | Acknowledged; |

**Description**

`drawFinalNumberAndMakeLotteryClaimable` performs multiple distinct operations, break it down into smaller internal functions. Each function should have a single responsibility.

**Exploit Scenario**

N/A

**Recommendations**

```
function drawFinalNumberAndMakeLotteryClaimable() external {
    uint32 finalNumber = getFinalNumber();
    uint256[6] memory rewards = calculateRewards(finalNumber);
    updateLotteryStatusAndRewards(rewards);
}

function getFinalNumber() internal returns (uint32) {
    // Logic to get the final number
    // ...
}

function calculateRewards(uint32 finalNumber) internal returns (uint256[6] memory) {
    // Logic to calculate rewards
    // ...
}

function updateLotteryStatusAndRewards(uint256[6] memory rewards) internal {
    // Logic to update the lottery status and rewards
    // ...
}
```

**Results**

Pending;

3. **Lack of event emission for ticket price changes**

```
function setTicketPrice(uint256 _priceTicket) external onlyOwner {
    priceTicket = _priceTicket;
}
```

| Severity | **Enhancement** |
|---|---|
| Source | Powerball.sol#L363 |
| Commit | n/a; |
| Status | Acknowledged; |

**Description**

Setting the ticket price is related to the business of the platform therefore, it is better to inform users by emitting an event.

**Exploit Scenario**

N/A

**Recommendations**

Add event emit to this function therefore, the user will be informed about the status of the contract.

### Results

Pending;

4. **Lack of event emission for pause functions**

| Severity | **Enhancement** |
|---|---|
| Source | Powerball.sol#L359 |
| Commit | n/a; |
| Status | Acknowledged; |

### Description

The pause function is considered as important system configuration, which influences `startLottery()`.

```
function setPaused(bool flag) external onlyOwner {
    paused = flag;
}
```

### Exploit Scenario

N/A

### Recommendations

Add event emit to this function therefore, the user will be informed about the status of the contract.

### Results

Pending;

5. **Floating Pragma**

| Severity | **Enhancement** |
|---|---|
| Source | Global |
| Commit | n/a; |
| Status | Acknowledged; |

### Description

There are three different compilers used by the `powerball` contract and its related libraries. It is recommended to use a fixed solidity compiler version in the main contract file.

### Exploit Scenario

N/A

### Recommendations

Change solidity ˆ0.8.4 to one version (line 2).

### Results

Pending;

## Appendix I: Security Issue Severities

| Level of Severity | Explanation |
|---|---|
| High | Issues that have security vulnerabilities with high risks. A high level risk issue is defined as when a successfully exploit occurs, it results in direct loss of funds or permanent freezing of funds. The high risk severity issues must be resolved. |
| Medium | Issues that have security vulnerabilities with medium risks. A medium level risk issue is only exploitable under conditions or with some prerequisites. All medium severity issues must be resolved unless there are clear explanations or standard operation procedures to avoid the potential rish to be exploited. |
| Low | Issues that have security vulnerabilities with low risks. A low level risk issue would not result in the failure of the system when a hacker realizes it. The client can decide if they want to revise the code for a low level risk issue. |
| Informational | Issues that pose no risk to the system but related with the best practices. The client can decide if they want to revise the code for an informational level issue. |
| Concerned | Issues that are not included in the auditing scope but we suggest the client to double check before the launch. |

## Appendix II: Status Categories

| Status | Description |
|---|---|
| Unresolved | The issue is not acknowledged nor resolved. |
| Acknowledged | The issue is acknowledged but not resolved. |
| Resolved | The issue has been resolved. |

## Disclaimer

Canary Technologies Inc. receives compensation from the Client for performing the smart contract and auditing analysis contained in this report. The report is solely for the Client and published with their consent. The scope of our audit is limited to a review of code, and only the code we note as being within the scope of our audit. It is important to note that the Solidity code itself presents unique and unquantifiable risks since the language itself continues to be developed and is subject to unknown risks and flaws. Our sole goal is to help reduce the risk of security attacks that is inherent in utilizing new and consistently changing technologies. Thus, Canary Technologies Inc in no way claims any guarantee of security or functionality of the code we agree to analyze.

In addition, our reports do not provide any indication of the audited technologies' proprietors, business and/or legal compliance. Also, our reports do not provide investment advice and should not be used to make decisions about investment or involvement with any particular project. We reserve the right to distribute the reports through other means, including via Canary Technologies' publications. We may make the reports available to parties other than the Client (i.e., "third parties") or on our website in hopes that it can help the blockchain sector develop technical best practices.