# BRKT

# Audit Report

Mon Nov 11 2024

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

# BRKT Audit Report

## 1 Executive Summary

### 1.1 Project Information

| | |
|---|---|
| Description | BRKT is a group prediction market that supports platforms for competitions, news, culture, and more. |
| Type | DeFi |
| Auditors | ScaleBit |
| Timeline | Thu Oct 17 2024 - Thu Oct 31 2024 |
| Languages | Solidity |
| Platform | Others |
| Methods | Architecture Review, Unit Testing, Manual Review |
| Source Code | https://github.com/brkted/brkt-contracts<br>https://github.com/brkted/brkt-contracts-audit |
| Commits | https://github.com/brkted/brkt-contracts:<br>5884e2c9ec423e86e79746425f559d071a3a1ba2<br>https://github.com/brkted/brkt-contracts-audit:<br>0ec8dc1be2d0c0547c6909867eb23c1400069dff<br>6fc069ea71fdbe81142ceab5bfe9402a6784e913 |

## 1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

| ID | File | SHA-1 Hash |
|---|---|---|
| CFA | src/competition/CompetitionFactory.sol | 401648c6c518120787bc834d32e018f0a3e15557 |
| PCS | src/competition/predictable/PredictableCompetitionState.sol | dbecdf223163861590eeae95c0c5e72d2c587c93 |
| PPCS | src/competition/predictable/PaidPredictableCompetitionState.sol | 974483e5b29f9e37c7fe9619cf23ea400c48ce70 |
| PPC | src/competition/predictable/PaidPredictableCompetition.sol | bb9555798aaf0c49b5bc097579a52fbe664d4a33 |
| PCO | src/competition/predictable/PredictableCompetition.sol | ec35dd296d240cf466db3dc782ff5e858b455266 |
| CRO | src/competition/CompetitionRouter.sol | a24ac8b89bad5a15716f58cedaf337b180c32444 |
| CST | src/competition/base/CompetitionState.sol | 3c89745fc1a1696630921db4678c6b6a88365816 |
| COM | src/competition/base/Competition.sol | 361f96cfaded9dff160f0e6b797985f695fe7438 |

## 1.3 Issue Statistic

| Item | Count | Fixed | Acknowledged |
| --- | --- | --- | --- |
| Total | 9 | 5 | 4 |
| Informational | 2 | 0 | 2 |
| Minor | 5 | 3 | 2 |
| Medium | 1 | 1 | 0 |
| Major | 0 | 0 | 0 |
| Critical | 1 | 1 | 0 |

## 1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

## 1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

### (1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

### (2) Code Review

The code scope is illustrated in section 1.2.

### (3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

# 2 Summary

This report has been commissioned by BRKT to identify any potential issues and vulnerabilities in the source code of the BRKT smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 9 issues of varying severity, listed below.

| ID | Title | Severity | Status |
| --- | --- | --- | --- |
| CFA-1 | `CompetitionImpl` Type Is Set Incorrectly | Medium | Fixed |
| CFA-2 | CREATE3 is Not Available in The zkSync Era | Informational | Acknowledged |
| CFA-3 | Lack of Events Emit | Informational | Acknowledged |
| COM-1 | `_initializeCompetition()` Function is Missing Duplicate Status Check | Minor | Acknowledged |
| COM-2 | `completeMatch` Function Manually Sets the Result | Minor | Acknowledged |
| PCO-1 | `createBracketPrediction` Function is Missing Origin Validation | Critical | Fixed |
| PCO-2 | `_saveUserPrediction` Function Added Repeatedly | Minor | Fixed |
| PPC-1 | Use of Unsafe ERC20 Transfer Functions | Minor | Fixed |
| PPC-2 | Code Optimization | Minor | Fixed |

# 3 Participant Process

Here are the relevant actors with their respective abilities within the BRKT Smart Contract :

**Owner**

- The owner can call the `setProtocolFee` to set the `protocolFee` .
- The owner can call the `claimProtocolFees` to collect the fee.

**User**

- Users can call the `createCompetition` `createPredictableCompetition` or `createPaidPredictableCompetition` function to create a competition.
- Users can call the `createBracketPrediction` function to create bracket prediction.
- Users can call the `claimRewards` function to claim the reward.

# 4 Findings

## CFA-1 `CompetitionImpl` Type Is Set Incorrectly

Severity: Medium

Status: Fixed

Code Location:

src/competition/CompetitionFactory.sol#87

Descriptions:

When the `createPredictableCompetition` function is called to create a contract of type `PredictableCompetition`, the `impl` is incorrectly set to `BASE`.

Suggestion:

It is recommended that this be changed to the correct type.

# CFA-2 CREATE3 is Not Available in The zkSync Era

Severity: Informational

Status: Acknowledged

Code Location:

src/competition/CompetitionFactory.sol#61,88,123

Descriptions:

the logic to compute the address of Create2 is different from Ethereum, as shown below, so the CREATE3 library cannot be used as it is.

The description of CREATE and CREATE2 (https://docs.zksync.io/build/developer-reference/ethereum-differences/evm-instructions#create-create2) states that Create cannot be used for arbitrary code unknown to the compiler.

This cause registry returns an incorrect `preCalculatedAddress`, causing the anchor to be registered to an address that is not the actual deployed address.

Suggestion:

According to the description in the document (https://github.com/0xsequence/create3/blob/master/contracts/Create3.sol), it is recommended to confirm the supporting chain when deploying multiple chains, and because the predicted address may be occupied in advance (according to the feature Same contract addresses on different EVM networks), it is necessary to confirm the empty address when deploying.

# CFA-3 Lack of Events Emit

Severity: Informational

Status: Acknowledged

Code Location:

src/competition/CompetitionFactory.sol#153;

src/competition/CompetitionRouter.sol#64;

src/competition/base/Competition.sol#49

Descriptions:

The contract lacks appropriate events for monitoring operations, such as `setProtocolFee()`, `setFactory()` functions which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for the function.

## COM-1 `_initializeCompetition()` Function is Missing Duplicate Status Check

Severity: Minor

Status: Acknowledged

Code Location:

src/competition/base/Competition.sol#113

Descriptions:

The permission check in the `_initializeCompetition()` function checks whether the caller is the `competitionFactory` address. `factory` is controlled by the `owner` permission. In the extreme case of a single point of account failure, the initialization function may be called again. However, these checks do not prevent the initialize function from being called repeatedly. Normally, the initialization function of a contract should contain a state variable to mark whether the contract has been initialized to prevent repeated calls.

Suggestion:

To fix this, you can add a boolean state variable initialized and check it in the initialize function:

```
bool private initialized = false;

function initialize(
    address _competitionOwner,
    string calldata _competitionName,
    uint16 _numTeams,
    uint64 _startingEpoch,
    uint64 _expirationEpoch,
    string[] memory _teamNames,
    string memory _bannerURI
) external override {
    require(!initialized, "Contract already initialized");
    initialized = true;
    _initializeCompetition(
        _competitionOwner, _competitionName, _numTeams, _startingEpoch,
_expirationEpoch, _teamNames, _bannerURI
```

```
    );
}
```

## COM-2 `completeMatch` Function Manually Sets the Result

Severity: Minor

Status: Acknowledged

Code Location:

src/competition/base/Competition.sol#64

Descriptions:

For example, when setting the result, the winning team is "1 3 2".

```
  function completeMatch(uint256 _matchId, uint8 _winningTeamId) external onlyOwner
whenInProgress {
    _completeMatch(_matchId, _winningTeamId);
  }
```

Suggestion:

It is recommended to add checks to prevent incorrect setup.

## PCO-1 `createBracketPrediction` Function is Missing Origin Validation

Severity: Critical

Status: Fixed

Code Location:

src/competition/predictable/PredictableCompetition.sol#55

Descriptions:

The `createBracketPrediction` function does not check whether it is the user who made the prediction. Others can call this function to change the original user's prediction results.

```
  function createBracketPrediction(address _registrant, uint8[] calldata
_matchPredictions)
      public
      virtual
      whenNotLive
  {
      _saveUserPrediction(_registrant, _matchPredictions);
  }
```

Suggestion:

It is recommended that after the User sets it, only the User can modify it.

# PCO-2 `_saveUserPrediction` Function Added Repeatedly

Severity: Minor

Status: Fixed

Code Location:

src/competition/predictable/PredictableCompetition.sol#174

Descriptions:

`matchPredictionsToUser` variable added repeatedly.

```
        if (!hasBracket) {
            matchPredictionsToUser[i][_matchPredictions[i]].add(_user);
            _addUserMatchPrediction(_user, i, _matchPredictions[i]);
            continue;
        }
```

Suggestion:

It is recommended to remove duplicate additions.

# PPC-1 Use of Unsafe ERC20 Transfer Functions

Severity: Minor

Status: Fixed

Code Location:

src/competition/predictable/PaidPredictableCompetition.sol#276 295 308;

src/interfaces/ICompetitionRouter.sol#49

Descriptions:

In contracts, including PaidPredictableCompetition and CompetitionRouter, unsafe `transfer()` and `transferFrom()` functions are used for ERC20 token transfers. Standard `transfer()` and `transferFrom()` functions may return false instead of reverting in some token contracts, or not return anything when failing. This can lead to silent failures, preventing the calling contract from properly handling transfer failures.

```
IERC20(registrationFeeInfo.paymentToken).transfer(msg.sender, pendingRewards);
```

Suggestion:

It is recommended to Replace all `transfer()` and `transferFrom()` calls with `safeTransfer()` and `safeTransferFrom()` functions from OpenZeppelin's SafeERC20 library.

# PPC-2 Code Optimization

Severity: Minor

Status: Fixed

Code Location:

src/competition/predictable/PaidPredictableCompetition.sol#323

Descriptions:

In the code, `pendingRewards_` can use `if/else` to optimize the code execution process.

```
  function calculatePendingRewards(address _user) public view override returns
(uint256 pendingRewards_) {
    // Don't calculate pending rewards if the user has already claimed them, or if the
competition hasn't finished yet
    if (!claimedRewards[_user] && hasFinished) {
      uint256 percentOfTotal = _getUserScorePercent(_user);
      // percentOfTotal is a number between 0 and 1e6, so we divide by 1e6 to get the
relative amount of token
      pendingRewards_ = totalRegistrationReserves * percentOfTotal / 1e6;
      if (competitionFactory.protocolFee() > 0) {
        pendingRewards_ = mulDiv(totalRegistrationReserves, 1e6 -
competitionFactory.protocolFee(), 1e6)
          * percentOfTotal
      }
    }
  }
```

Suggestion:

It is recommended to use `if/else` to optimize and save gas.

# Appendix 1

## Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

## Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

# Appendix 2

## Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.