Three Sigma

Code Audit

SCOREPLAY

Scoreplay Prediction Market

Disclaimer

Code Audit

Scoreplay Prediction Market

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Code Audit

Scoreplay Prediction Market

Table of Contents

Summary

Code Audit

Scoreplay Prediction Market

Summary

Three Sigma audited Scoreplay in a 0.8 person week engagement. The audit was conducted from 02/04/2025 to 03/04/2025.

Protocol Description

Scoreplay is a decentralized prediction market built on Basecamp and Sophon, enabling users to place secure, transparent bets on a wide range of supported sports events. Smart contracts ensure trustless payouts and fair play, giving bettors full control over their funds and bets.

Scope

Code Audit

Scoreplay Prediction Market

Scope

Filepath	nSLOC
EnhancedPredictionMarket.sol	248
Total	248

Assumptions

OpenZeppelin library is considered secure.

Methodology

Code Audit

Scoreplay Prediction Market

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit, we classify findings based on Immunefi's Vulnerability Severity Classification System (v2.3) as a guideline. The final classification considers both the potential impact of an issue, as defined in the referenced system, and its likelihood of being exploited. The following table summarizes the general expected classification according to impact and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Impact / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

Scoreplay Prediction Market

Project Dashboard

Application Summary

Name	Scoreplay
Repository	https://github.com/ScorePlay-xyz/sports-contract-audit
Commit	4cdff28
Language	Solidity
Platform	Basecamp, Sophon

Engagement Summary

Timeline	02/04/2025 to 03/04/2025
№ of Auditors	2
Review Time	0.8 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	1	1	0
Medium	2	2	0
Low	2	2	0
None	4	3	1

Category Breakdown

Suggestion	4
Documentation	0
Bug	5
Optimization	0
Good Code Practices	0

Risk Section

Code Audit

Scoreplay Prediction Market

Risk Section

No risks identified.

Findings

Code Audit

Scoreplay Prediction Market

Findings

3S-Scoreplay-H01

Lack of frontrunning protection in **`resolveCondition`** allows unfair advantage

Id	3S-Scoreplay-H01
Classification	High
Impact	High
Likelihood	Medium
Category	Bug
Status	Addressed in #ad67b0b.

Description

The **`EnhancedSportsPrediction::resolveCondition`** function can be frontrun by malicious users. Since there is no check for **`endTime`**, a user can monitor the mempool for a **`resolveCondition`** transaction, extract the **`winningOutcome`**, and place a large bet on it before the condition is resolved.

Recommendation

Divide the **`resolveCondition`** logic into two separate functions: one to block trading and another to set the winning outcome.

3S-Scoreplay-M01

Global house fee modification affects existing bets

Id	3S-Scoreplay-M01
Classification	Medium
Impact	Medium
Likelihood	High
Category	Bug
Status	Addressed in #de4a663.

Description:

The **`EnhancedSportsPrediction::setHouseFee`** function allows the owner to change the global house fee, which affects all existing bets. This can lead to unexpected changes in payout calculations for users who have already placed bets.

More critically, the house fee calculation occurs in two separate places (**`EnhancedSportsPrediction::resolveCondition`** and **`EnhancedSportsPrediction::claimPayout`**), creating a potential inconsistency when the fee changes between these operations.

This creates two significant impacts:

1. Users receive less than expected: If the house fee increases between condition resolution and payout claiming, users will receive lower payouts than they should based on the fee at the time of resolution.

2. Funds get trapped in the contract: The inconsistency creates a situation where **`totalFeesCollected`** doesn't match the actual fees collected from users. This breaks the economic integrity of the protocol and can result in funds permanently locked in the contract.

Recommendation:

Extend the **`Condition`** struct to host a houseFee field. Store the house fee within each **`Condition`** struct at the time of creation and use this stored fee for payout calculations in **`resolveCondition`** an **`claimPayout`** functions.

3S-Scoreplay-M02

Lack of reimbursement mechanism for incorrect bets leads to fund locking

Id	3S-Scoreplay-M02
Classification	Medium
Impact	High
Likelihood	Low
Category	Bug
Status	Addressed in #dd02487.

Description

In the **`EnhancedSportsPrediction::claimPayout`** function, if all users bet on the wrong outcome, the funds remain locked in the contract due to the line **`if (outcomePool == 0) revert NoBetsOnOutcome();`**. Currently there is no mechanism to reimburse users or collect the funds as house fees.

Recommendation

In such situation, either collect the funds as house fees or implement a mechanism to reimburse users with their bets (possibly minus the house fee).

3S-Scoreplay-L01

Lack of outcome range validation in **`createCondition`** allows invalid bets

Id	3S-Scoreplay-L01
Classification	Low
Impact	Medium
Likelihood	Low
Category	Bug
Status	Addressed in #cb8673c.

Description

The **`EnhancedSportsPrediction::createCondition`** function allows the creation of a betting condition without specifying a valid range of possible outcomes. This omission can lead to invalid bets being placed, as there is no mechanism to validate the **`outcome`** parameter in **`EnhancedSportsPrediction::placeBet`**.

Recommendation

To improve the user experience, it is suggested to implement a range of possible outcomes in the **`Condition`** struct and enforce this range in the **`EnhancedSportsPrediction::placeBet`** function.

3S-Scoreplay-L02

Use safe transfer for ERC20 tokens

Id	3S-Scoreplay-L02
Classification	Low
Impact	Low
Likelihood	Medium
Category	Bug
Status	Addressed in #0542a25.

Description

The protocol uses direct ERC20 token transfer functions for the collateral token but doesn't implement proper safety checks for non-standard tokens. The contract assumes all collateral tokens correctly return a boolean value from their **`transfer`** and **`transferFrom`** functions, which isn't always the case.

Some tokens don't follow the EIP20 standard correctly and return void instead of a boolean. When calling these non-compliant tokens, the transaction will revert due to the failed return value check.

Recommendation

Use OpenZeppelin's SafeERC20 library which handles both standard and non-standard token implementations.

3S-Scoreplay-N01

Absence of minimum match duration in **`createCondition`**

Id	3S-Scoreplay-N01
Classification	None
Category	Suggestion
Status	Acknowledged

Description

The **`EnhancedSportsPrediction::createCondition`** function does not enforce a minimum duration for the betting period. This oversight can lead to conditions being created with an **`endTime`** that is too close to the current time, potentially ending the betting period immediately and affecting user experience.

Recommendation

Enforce a minimum duration for the betting period in the **`EnhancedSportsPrediction::createCondition`** function to protect from human errors.

3S-Scoreplay-N02

CEI Pattern Violation in **`placeBet`** and **`claimPayout`** Functions

Id	3S-Scoreplay-N02
Classification	None
Category	Suggestion
Status	Addressed in #0d9f36f.

Description

The **`EnhancedSportsPrediction::placeBet`** and **`EnhancedSportsPrediction::claimPayout`** functions do not properly follow the CEI (Checks-Effects-Interactions) pattern, which is a crucial best practice in Solidity for preventing reentrancy attacks and other potential vulnerabilities.

While both functions use the **`nonReentrant`** modifier which provides some protection, not following CEI pattern doesn't adhere to best practices and could potentially introduce subtle edge cases or read-only reentrancy vulnerabilities.

Recommendation

Adopt the CEI pattern more rigorously in both functions:

- Checks: Verify conditions and calculate amounts.

- Effects: Update all relevant state variables.

- Interactions: Only then perform token transfers or other external calls.

3S-Scoreplay-N03

`collateralToken` variable should be declared as immutable

Id	3S-Scoreplay-N03
Classification	None
Category	Suggestion
Status	Addressed in #c17a69d.

Description

In the **`EnhancedSportsPrediction`** contract, the **`collateralToken`** variable is used to store the address of the ERC20 token that is used for placing bets. This variable is set during the contract's construction and is never modified afterward. Since the **`collateralToken`** address is immutable after deployment, it should be declared as **`immutable`** to save gas during operations.

Recommendation

Declare the **`collateralToken`** variable as **`immutable`** to optimize gas usage.

```
- IERC20 public collateralToken;
+ IERC20 public immutable collateralToken;
```

3S-Scoreplay-N04

Lack of pagination in **`getUserMatches`** and **`getUserPayouts`** leads to potential block gas limit issues

Id	3S-Scoreplay-N04
Classification	None
Category	Suggestion
Status	Addressed in #2d3a3d2.

Description

The **`EnhancedSportsPrediction::getUserMatches`** and **`EnhancedSportsPrediction::getUserPayouts`** functions return arrays of match IDs that a user has participated in or claimed payouts for, respectively. As the number of matches grows, these arrays can become large enough to exceed the block gas limit, making it impossible to retrieve all data in a single call. This limitation can prevent users from accessing their complete betting history, affecting user experience and data accessibility.

Recommendation

Implement optional pagination for these functions using function overloading. This allows users to specify a range of indices to retrieve, ensuring that data can be accessed even if it grows significantly. Implementing the solution through function overloading allows the current functions to be kept.