QuillAudits

AUDIT REPORT

September 2025

For

Fairplay

Table of Content

Executive Summary 03

Number of Security Issues per Severity 05

Summary of Issues 06

Checked Vulnerabilities 07

Techniques and Methods 09

Types of Severity 11

Types of Issues 12

Severity Matrix 13

Medium Severity Issues 14

1. Centralization Risk 14

Low Severity Issues 15

2. Lack of Replay Protection 15

Informational Issues 16

3. Use of TwoStepOwnable for Ownership Transfers 16

4. Floating Solidity Pragma Version 16

Functional Tests 17

Automated Tests 17

Threat Model 18

Notes for Users 19

Closing Summary & Disclaimer 19

Executive Summary

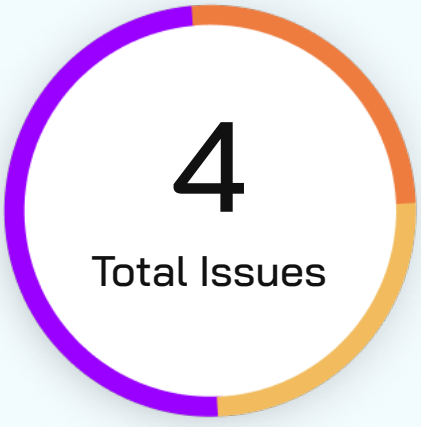
Project Name	FairPlay
Protocol Type	Gaming
Project URL	https://fairplay-git-feature-chart-worldsdotfuns-projects.vercel.app/
Overview	FairPlay Game Contract is an upgradeable smart contract for managing a provably fair gaming platform on Abstract L2 (zkSync Era). Players can create games by depositing ETH bets, with each game tracked by a unique ID and cryptographic seed hash for fairness verification. The contract supports two outcomes: 1) players can either cash out winnings or have their game marked as lost, both requiring server signatures for authorization. 2)Admin addresses can perform privileged operations like fund withdrawals and managing game states without signatures.
Audit Scope	The scope of this Audit was to analyze the FairPlay Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/Aark-AI/fairplay-contracts/blob/main/contracts/FairPlay.sol
Contracts in Scope	contracts/FairPlay.sol
Commit Hash	0ff4d37b13cf4fe09fd1dbe4109e60cec6d24ce3
Branch	Main
Language	Solidity
Blockchain	Ethereum
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	4th September 2025 - 12th September 2025
Updated Code Received	12th Sep 2025

Review 2	12th Sep 2025
Fixed In	be721bd82e6d6305d41e2752944fc05e395692c1

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity



Critical	0 (0%)
High	0 (0%)
Medium	1 (25%)
Low	1 (25%)
Informational	2 (50%)

Severity

Issues	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	0	**1**	**1**	0
Partially Resolved	0	0	0	0	0
Resolved	0	0	0	0	**2**

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Centralization Risk	Medium	Acknowledged
2	Lack of Replay Protection	Low	Acknowledged
3	Use of TwoStepOwnable for Ownership Transfers	Informational	Resolved
4	Floating Solidity Pragma Version	Informational	Resolved

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Computations Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls

- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw
- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code
- Use of best practices
- Code documentation and comments, match logic and expected behavior
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper
- Implementation of ERC standards
- Efficient use of gas
- Code is safe from re-entrancy and other vulnerabilities

The following techniques, methods, and tools were used to review all the smart contracts:

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms Used for Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity Static Analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are five levels of severity, and each of them has been explained below.

Critical: Immediate and Catastrophic Impact

Critical issues are the ones that an attacker could exploit with relative ease, potentially leading to an immediate and complete loss of user funds, a total takeover of the protocol's functionality, or other catastrophic failures. Critical vulnerabilities are non-negotiable; they absolutely must be fixed.

High (H): Significant Risk of Major Loss or Compromise

High-severity issues represent serious weaknesses that could result in significant financial losses for users, major malfunctions within the protocol, or substantial compromise of its intended operations. While exploiting these vulnerabilities might require specific conditions to be met or a moderate level of technical skill, the potential damage is considerable. These findings are critical and should be addressed and resolved thoroughly before the contract is put into the Mainnet.

Medium (M): Potential for Moderate Harm Under Specific Circumstances

Medium-severity bugs are loopholes in the protocol that could lead to moderate financial losses or partial disruptions of the protocol's intended behavior. However, exploiting these vulnerabilities typically requires more specific and less common conditions to occur, and the overall impact is generally lower compared to high or critical issues. While not as immediately threatening, it's still highly recommended to address these findings to enhance the contract's robustness and prevent potential problems down the line.

Low (L): Minor Imperfections with Limited Repercussions

Low-severity issues are essentially minor imperfections in the smart contract that have a limited impact on user funds or the core functionality of the protocol. Exploiting these would usually require very specific and unlikely scenarios and would yield minimal gain for an attacker. While these findings don't pose an immediate threat, addressing them when feasible can contribute to a more polished and well-maintained codebase.

Informational (I): Opportunities for Improvement, Not Immediate Risks

Informational findings aren't security vulnerabilities in the traditional sense. Instead, they highlight areas related to the clarity and efficiency of the code, gas optimization, the quality of documentation, or adherence to best development practices. These findings don't represent any immediate risk to the security or functionality of the contract but offer valuable insights for improving its overall quality and maintainability. Addressing these is optional but often beneficial for long-term health and clarity.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Severity Matrix

Impact

Likelihood	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

Medium Severity Issues

Centralization Risk

Acknowledged

Path

FairPlay.sol

Description

The contract relies on a set of admin addresses for critical operations (e.g., validating server signatures to create games). This introduces a centralization risk, as these privileged accounts can unilaterally control the game creation process and potentially manipulate outcomes. Compromise of any admin key could lead to malicious game creation, blocking legitimate games, or denial of service.

Impact

If the admin keys are compromised or misused, it could allow unauthorized creation or manipulation of games, harming the integrity and fairness of the platform.

Recommendation

Implement multisignature authorization for critical actions.Consider introducing community governance or on-chain verification to reduce reliance on centralized authority.

FairPlay Team's Comment

Even if the server admin key is compromised, it will not cause problems for users' funds. We assess this particular risk to be lower and have decided to securely manage and store the admin key using methods like a Privy server wallet. Furthermore, since a signature is required for the creation of every game, it is a practical necessity for the server signature to be centralized. We view this as an unavoidable structural aspect rather than a risk and have decided not to change the code.

Low Severity Issues

Lack of Replay Protection

Acknowledged

Path

FairPlay.sol

Function Name

`createGame`
`cashOut`
`markGameAsLost`

Description

The contract verifies admin authorization using ECDSA signatures but does not include any replay protection (i.e. nonce). This allows any valid signed message from an admin to be submitted multiple times by the same player or by others (if they get access to the signed payload) as long as the deadline has not passed.

Impact

Replay could allow unauthorized repeated execution of sensitive actions like payouts if state does not inherently block it.

Likelihood

Any leaked or reused admin signature can be exploited until expiry.

Recommendation

Include a nonce in the signed data.

FairPlay Team's Comment

If the ECDSA signature for createGame were to be stolen, the use of same preliminaryGameId would cause a GameAlreadyExists error, preventing a replay attack.
For cashOut and markGameAsLost, we create the signature by including the onChainGameId. This restricts the signature's validity to a single, specific game. Consequently, it's impossible to reuse the signature for a different game, and it cannot be replayed within the same game either.
For these reasons, we do not consider this a risk and do not plan to modify the code.

Informational Issues

Use of TwoStepOwnable for Ownership Transfers

Resolved

Path

FairPlay.sol

Description

The contract uses OpenZeppelin's TwoStepOwnable pattern for ownership transfers.
This is a good practice as it prevents accidental loss of ownership by requiring the new owner to explicitly accept ownership after it has been transferred.
This is not a vulnerability, but it's noted here for completeness.

Recommendation

Keep using TwoStepOwnable as it enhances security during ownership transfers.

Floating Solidity Pragma Version

Resolved

Path

FairPlay.sol

Description

The contract uses a floating Solidity version pragma (^0.8.24).
This can lead to unexpected behavior or incompatibility if a newer compiler version with breaking changes or undiscovered bugs is used during compilation

Recommendation

Use a fixed Solidity version pragma (e.g. pragma solidity 0.8.24;) to ensure consistent compilation and bytecode.

Functional Tests

Some of the tests performed are mentioned below:

- Game Creation Verified that a valid createGame call with a correct admin signature successfully creates a new game.
- Duplicate Game Prevention Ensured that attempting to create a game with an already used preliminaryGameId correctly reverts.
- Signature Expiry Checked that a createGame transaction reverts if block.timestamp > deadline.
- Invalid Signature Confirmed that calls with invalid or tampered serverSignature revert as expected.
- Fund Handling Verified that the msg.value sent during createGame is correctly recorded and processed.
- Admin Access Control Tested that only addresses marked as isAdmin can generate valid signatures.
- Ownership Transfer Validated that ownership can be transferred and only the new owner can perform owner-only actions.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Threat Model

Contract	Function	Threats
FairPlay	createGame	duplicate ID, expired, zero value, long ID **Admin signatures required centralized control over game creation.**
FairPlay	cashOut	access control, expired, zero payout, replay, **Admin can execute payouts centralized control over fund distribution.**
FairPlay	markGameAsLost	access control, expired, already finished, **Admin can mark games lost centralized control over outcomes.**
FairPlay	_verifyAnyAdminSignature	non-admin, empty sig **Relies on admin keys**
FairPlay	addAdmin / removeAdmin:	**Only owner can manage admins centralized governance.**
FairPlay	withdrawFunds	**Only owner can withdraw**

Notes for Users

- Do not treat the game as fully decentralized. It relies on trusted admins and servers.
- Check official announcements for the current owner and admin addresses; stale admins may still have privileges.
- Be aware of potential upgradeability. New versions may alter trust dynamics.
- Understand downtime risks. If the backend server is unavailable, players may not be able to finalize or withdraw from games until operators intervene.

Closing Summary

In this report, we have considered the security of FairPlay. We performed our audit according to the procedure described above.

Medium,Low,Informational found during the Audit. FairPlay team resolved a few and acknowledged other issues

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With seven years of expertise, we've secured over 1400 projects globally, averting over $3 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

7+ Years of Expertise	**1M+** Lines of Code Audited
50+ Chains Supported	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

September 2025

For

Fairplay

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com