






AUDIT REPORT

September 2025

For

 **Fairplay**

Table of Content

Executive Summary	03
Number of Security Issues per Severity	04
Summary of Issues	05
Checked Vulnerabilities	06
Techniques and Methods	07
Types of Severity	08
Types of Issues	09
Severity Matrix	10
 High Severity Issues	11
1. Indirect Object Reference leads to referrer's data leakage	11
 Medium Severity Issues	12
2. Authorization token not enforced on GET Requests	12
3. Improper Authorization Validation & Lack of Rate Limiting on POST /game/faucet	13
 Low Severity Issues	15
4. Clickjacking	15
Closing Summary & Disclaimer	16



Executive Summary

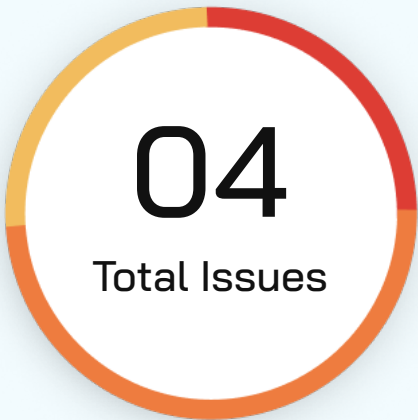
Project Name	Fairplay
Protocol Type	Dapp Pentest
Project URL	https://fairplay-git-feature-chart-worldsdotfun-projects.vercel.app/
Overview	Fairplay is a tap-to-trade game that turns trading into a fun, social, and fair experience. Players click on live chart interfaces to trade, earning points by competing on global leaderboards.
Review 1	6th Sept 2025 - 12th Sept 2025
Updated Code Received	17th September 2025
Review 2	19th September 2025

Verify the Authenticity of Report on QuillAudits Leaderboard:

<https://www.quillaudits.com/leaderboard>



Number of Issues per Severity



Critical	0(0.0%)
High	1 (25.0%)
Medium	2 (50.0%)
Low	1 (25.0%)
Informational	0(0.0%)

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



Summary of Issues

Issue No.	Issue Title	Severity	Status
1	IDOR leading to Other user's Data Disclosed	High	Resolved
2	AUTHORIZATION BEARER Undefined in operations	Medium	Resolved
3	JWT Token does not expire	Medium	Resolved
4	Clickjacking	Low	Resolved



Checked Vulnerabilities

✓ Improper Authentication

✓ Improper Resource Usage

✓ Improper Authorization

✓ Insecure File Uploads

✓ Insecure Direct Object References

✓ Client-Side Validation Issues

✓ Rate Limit

✓ Input Validation

✓ Injection Attacks

✓ Cross-Site Scripting (XSS)

✓ Cross-Site Request Forgery

✓ Security Misconfiguration

✓ Broken Access Controls

✓ Insecure Cryptographic Storage

✓ Insufficient Cryptography

✓ Insufficient Session Expiration

✓ Insufficient Transport Layer Protection

✓ Unvalidated Redirects and Forwards

✓ Information Leakage

✓ Broken Authentication and Session Management

✓ Denial of Service (DoS) Attacks

✓ Malware

✓ Third-Party Components

And More..



Techniques and Methods

Throughout the pentest of application, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest:

Burp Suite

DNSenum

Dirbuster

SQLMap

Netcat

Acunetix

Neucli

Nabbu

Turbo Intruder

Nessus

Nmap

Metasploit

Horusec

Postman

And Many more..



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		
		High	Medium	Low
Likelihood	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.



High Severity Issues

Indirect Object Reference leads to referrer's data leakage

Resolved

Description

The API endpoint `/users/{id}` is vulnerable to Insecure Direct Object Reference (IDOR). By modifying the `did:privy:<USER_ID>` value in the request URL, an attacker can retrieve sensitive details of other users/referrers without authorization checks. This exposes internal referral codes, wallet addresses, usernames, and related account metadata,

Vulnerable URL

GET `https://dev-api.fairplay.trade/users/did:privy:<USER_ID>`

Impact

- Sensitive Data Disclosure: Attackers can enumerate and extract wallet addresses, referral codes, usernames, and account activity timestamps of other users.
- Privacy Violation: Exposure of personal or financial identifiers violates user privacy and could aid in phishing or targeted attacks.
- Reputation Risk: Trust in the platform diminishes if user referral/wallet information can be freely accessed.

Recommendation

- Implement Access Controls: Validate that the requesting user is authorized to access the requested user ID before returning any data.
- Use Indirect Identifiers: Replace direct user IDs in URLs with opaque, non-sequential references that cannot be guessed.

POC

```
GET /users/did:privy:cmff7gpc4004kx0bzvhdhs6f HTTP/2
Host: dev-api.fairplay.trade
Authorization: Bearer <valid_token>
```

Exploit

Change the value after `did:privy:` to another valid user ID (e.g., `did:privy: <REFERRER_ID>`).
Observe that data for the chosen user is disclosed without authorization.



Medium Severity Issues

Authorization token not enforced on GET Requests

Resolved

Description

The GET endpoints responds successfully even when the supplied Authorization bearer token is not validated for the requested resource. The service returns user data for arbitrary did:privy:<ID> values without enforcing that the caller is authorized to view that resource. In other words the API accepts a bearer token but does not verify token validity/ownership/claims before returning sensitive user information.

This is an authentication/authorization failure (missing or ineffective server-side token validation) that enables unauthenticated or unauthorized data disclosure.

Impact

- Unauthorized data disclosure: attacker can enumerate other users and retrieve referral IDs, wallet addresses, usernames and timestamps.
- Privacy breaches: personal/financial identifiers exposed (wallet addresses, referral data).
Account enumeration: enables mapping of platform users for follow-up attacks (phishing, blockchain targeting).
- Regulatory / compliance risk: exposure of PII could trigger legal/regulatory consequences depending on jurisdiction/data classification.

Recommendation

- Enforce server-side JWT validation for every request (PATCH and GET both)
Verify token signature using the correct public key / secret.
Check exp, iat, nbf claims and reject expired/invalid tokens.
- Validate token audience and issuer
Confirm aud matches the API audience and iss is the expected issuer.
- Do not rely on client-side protections or CORS
CORS is not an authentication control. All access checks must occur server-side.



Improper Authorization Validation & Lack of Rate Limiting on POST /game/faucet

Resolved

Description

The POST/game/faucet endpoint accepts requests where the Authorization header value is the literal string Bearer undefined and still processes the request. Additionally, there is no effective rate limiting or anti-abuse control on this endpoint, allowing an attacker or automated client to repeatedly call the faucet endpoint (e.g., 100+ requests) and drain rewards or consume system resources. Together, these issues allow unauthenticated or weakly authenticated callers to abuse the faucet functionality at scale.

Vulnerable URL

POST <https://dev-api.fairplay.trade/game/faucet>

Impact

- Unauthorized actions: Calls with Authorization: Bearer undefined are treated as authorized (or at least accepted) enabling unauthenticated users to trigger faucet transfers.
- Resource abuse / Fraud: No rate limiting means an attacker can mass-request tokens / rewards (financial loss, token inflation, exhaustion of funded resources).
- Financial loss & chain risk: Repeated faucet payouts to attacker-controlled addresses could cause financial impact and may facilitate downstream fraud or money laundering.
- Denial-of-service (DoS) or operational degradation: High request volumes may overload backend services (DB, blockchain relay, signing service).
- Audit & compliance risk: Actions made without real authentication/authorization break traceability and policy controls.

Recommendation

Always perform server-side token verification for protected endpoints

- Verify token signature (using appropriate secret or public key).
- Validate exp, nbf, iat, aud, iss and sub claims as applicable.
- Fail fast on malformed tokens

If the token cannot be parsed or verified, immediately return 401 Unauthorized and do not execute the endpoint logic.



POC

```
POST /game/faucet HTTP/2
Host: dev-api.fairplay.trade
Content-Length: 58
Sec-Ch-Ua-Platform: "macOS"
Authorization: Bearer undefined
Accept-Language: en-GB,en;q=0.9
Sec-Ch-Ua: "Chromium";v="139", "Not;A=Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/139.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://fairplay-git-feature-chart-worldsdotfuns-projects.vercel.app
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://fairplay-git-feature-chart-worldsdotfuns-projects.vercel.app/
Accept-Encoding: gzip, deflate, br
Priority: u=1, l
```

```
{"toAddress":"<HASH>"}
```



Low Severity Issues

Clickjacking

Resolved

Description

The site at <https://fairplay-git-feature-chart-worldsdotfuns-projects.vercel.app/> can be embedded inside an attacker-controlled iframe because the application does not consistently prevent framing. This enables a malicious page to visually hide or disguise the framed content and trick a user into clicking buttons or performing actions inside the framed application (UI redressing / clickjacking).

Impact

- Phishing / UX deception: An attacker can present the site inside a crafted page and trick authenticated users into taking actions they did not intend (e.g., clicking "Claim", "Transfer", approving wallet interactions).
- Unauthorized actions: If critical actions (payments, transfers, wallet binds) are available via a single click, users might unknowingly approve them while interacting with the attacker page.

Recommendation

Set a frame-ancestors CSP (preferred modern control)

Add the following HTTP response header (adjust allowed origins as needed):

POC

<https://clickjacker.io/test?url=https://fairplay-git-feature-chart-worldsdotfuns-projects.vercel.app/>



Closing Summary

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

Some issues of High, medium, low severity were found. Fairplay team resolved all the issues mentioned

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



Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillaudits.com