QuillAudits

AUDIT REPORT

January 2026

For

Kite

Table of Content

Executive Summary

Project Name	Kite Ai
Project URL	https://gokite.ai/
Overview	Gokite AI, also known as Kite AI, is a Layer 1 blockchain designed as the first AI payment blockchain, enabling autonomous AI agents to handle identity, payments, governance, and verification in an agentic economy
Audit Scope	The scope of this Audit was to perform a greybox pentest to analyze the Kite Ai domains and respective codebase for quality, security, and correctness.
Source Code Link	https://staging.kite.foundation https://github.com/gokite-ai/kite-foundation/tree/feat/security http://official.staging.gokite.ai https://kite.foundation/claim
Review 1	30th October 2025 - 2nd November 2025
Updated Code Received	23 December 2025
Review 2	30th December 2025 - 19th January 2026

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Detailed Scope

In-Scope Functionalities Tested

kite_website-feat-security Pages Tested (11 routes):

- `/` - Home/Landing page
- `/network` - Agentic Network page
- `/defi-suites` - DeFi Suites information page
- `/whitepaper` - Whitepaper page
- `/mica-whitepaper` - Mica Whitepaper page
- `/podcast` - Podcast page
- `/media` - Media page
- `/license` - License information page
- `/privacy` - Privacy policy page
- `/terms` - Terms of service page

kite-foundation-feat-security Pages Tested (6 routes):

- `/` - Home/Landing page
- `/about` - About page (route exists but may be commented/unused)
- `/privacy` - Privacy policy page
- `/terms` - Terms of service page
- `/tokenomics` - Tokenomics information page
- `/whitepaper` - Whitepaper page

External Route Tested:

- `https://kite.foundation/claim` - External claim page

Configuration Files

Files Analyzed:

- `next.config.js` / `next.config.ts` - Next.js configuration
- `package.json` - Dependency analysis
- `tsconfig.json` / `jsconfig.json` - TypeScript/JavaScript configuration

Number of Issues per Severity

Critical	0(0.0%)
High	1 (14.2%)
Medium	0 (0.0%)
Low	1 (14.2%)
Informational	5 (71.6%)

Severity

Issues	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	0	0	0	**5**
Partially Resolved	0	0	0	0	0
Resolved	0	**1**	0	**1**	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Clickjacking Can lead to theft of user's funds	High	Resolved
2	Missing Frame Protection leads to potential Clickjacking	Low	Resolved
3	Disclosure of Backend Framework via PoweredbyHeader	Informational	Acknowledged
4	Missing Essential Security Headers	Informational	Acknowledged
5	Outdated Nextjs version in kite_website-feat-security	Informational	Acknowledged
6	React StrictMode Disabled	Informational	Acknowledged
7	NPM Audit Report	Informational	Acknowledged

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	Acunetix	Nmap
DNSenum	Neucli	Metasploit
Dirbuster	Nabbu	Horusec
SQLMap	Turbo Intruder	Postman
Netcat	Nessus	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

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.

High Severity Issues

Clickjacking Can lead to theft of user's funds

Resolved

Vulnerable Endpoint

https://kite.foundation/claim

Description

The /claim page is vulnerable to clickjacking because the site currently allows third-party framing. An attacker can embed the claim page in an invisible iframe and overlay a deceptive UI that causes victims to click the real wallet Connect / Claim buttons inside the framed page. Because the claim flow includes wallet interactions that launch signature popups, a successful clickjacking chain could lead to token approvals or transfers that result in direct financial loss.

X-Frame-Options is legacy and not supported by the Content Security Policy (CSP) recommendation model. The modern and authoritative control is Content-Security-Policy: frame-ancestors. Use frame-ancestors as your primary defense; X-Frame-Options may still be added as a fallback for very old browsers but should not be relied on alone.

POC: Attack Scenario

1. Attacker creates a malicious website with an attractive fake airdrop page
2. Attacker embeds `https://kite.foundation/claim` in an iframe
3. Attacker overlays a semi-transparent deceptive page hiding the real site
4. Attacker positions fake "Connect Wallet" and "Claim" buttons exactly over the real buttons
5. User clicks what appears to be a legitimate airdrop button
6. Click actually triggers the real Connect Wallet or Claim action in the hidden iframe
7. User unknowingly connects wallet and approves transactions
8. Attacker can embed a button to transfer funds from the victim to attacker after a successful claim

POC

https://gist.github.com/Auditoooor/df035b4bf1649c813f8d2422e3548877

To Reproduce

1. Open above gist and save as attack.html and open in a web browser.
2. Observe that `https://kite.foundation/claim` loads in an iframe behind a semi-transparent overlay
3. Click the fake "CLAIM NOW" button positioned over the real Connect Wallet button
4. The click will trigger the real Connect Wallet action in the iframe
5. MetaMask popup will appear to connect the wallet

Impact

The vulnerability allows attacker to trick users into unintended wallet connection and Users may claim tokens thinking they're interacting with a different site whereas Legitimate tokens may be claimed on behalf of users without their awareness and subsequently theft of the tokens can be done by the attacker leading to loss of funds to the user.

Recommendation

Add the following headers in the **next.config.ts**

```
X-Frame-Options: DENY
Content-Security-Policy: frame-ancestors 'none';
```

Low Severity Issues

Missing Frame Protection leads to potential Clickjacking

Resolved

File Location

kite-foundation-feat-security/next.config.ts
kite_website-feat-security/next.config.js

Description

The current configuration defines a Content-Security-Policy (CSP) header, which includes a frame-src directive:

```
"frame-src 'self' *.gokite.ai;"
```

Current CSP contains frame-src 'self' *.gokite.ai; which only controls what the app may embed, not who may embed the app. The correct modern control to prevent other sites from embedding your pages is frame-ancestors in the CSP. X-Frame-Options is legacy/deprecated and should be considered a fallback only.

Impact

If successfully exploited, attackers can Trick authenticated users into unknowingly interacting with sensitive UI components.

Recommendation

Add a frame protection header to prevent the application from being embedded in external iframes.

You can address this by adding both legacy (X-Frame-Options) and modern (frame-ancestors) controls for stronger protection.

```
import type { NextConfig } from "next";

const securityHeaders = [
  {
    key: "X-Frame-Options",
    value: "DENY", // optional fallback header
  },
  {
    key: "Content-Security-Policy",
    value: [
      "default-src 'self' *.gokite.ai;",
      "script-src 'self' 'unsafe-inline' 'unsafe-eval' *.googletagmanager.com
*.gokite.ai;",
      "style-src 'self' data: 'unsafe-inline' *.gokite.ai https://
fonts.googleapis.com;",
      "img-src 'self' data: *.gokite.ai images.pexels.com;",
      "font-src 'self' data: *.gokite.ai https://fonts.gstatic.com;",
      "connect-src 'self' *.gokite.ai https://www.google-analytics.com;",
      "frame-src 'self' *.gokite.ai;",      // what this app can embed
      "frame-ancestors 'none';",            // prevents external framing
      "object-src 'none';",
      "base-uri 'self';",
      "form-action 'self' *.gokite.ai;",
    ].join(" "),
  },
];

const nextConfig: NextConfig = {
  poweredByHeader: false,
  async headers() {
    return [
      {
        source: "/:path*",
        headers: securityHeaders,
      },
    ];
  },
};

export default nextConfig;
```

Informational Issues

Disclosure of Backend Framework via PoweredbyHeader

Acknowledged

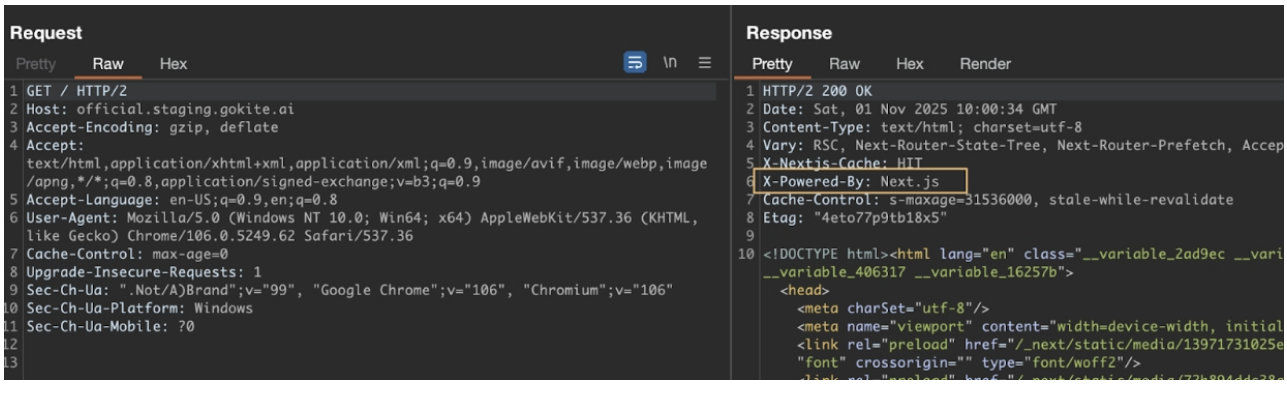
File Location

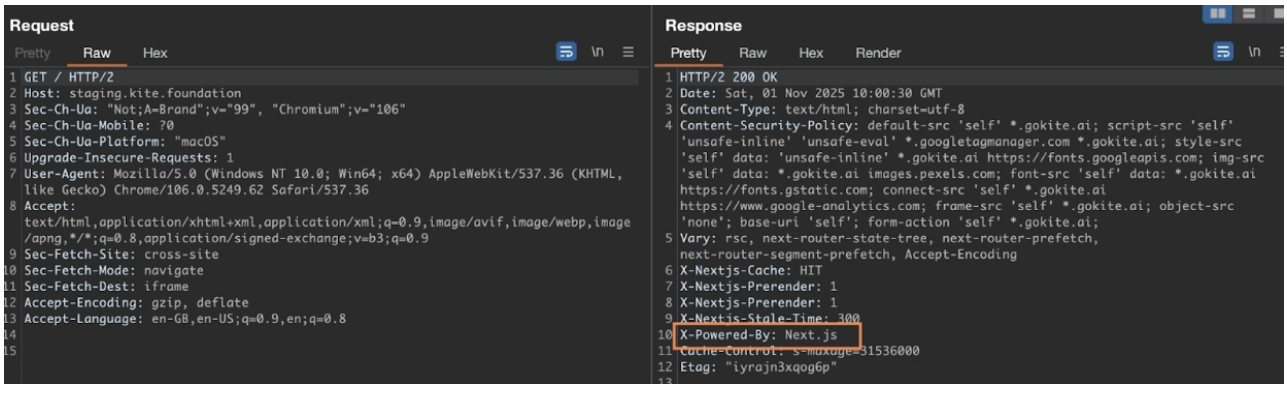
kite-foundation-feat-security/next.config.ts
kite_website-feat-security/next.config.js

Description

By default, Next.js includes an X-Powered-By HTTP response header that reveals the underlying backend framework (e.g., X-Powered-By: Next.js). This information disclosure may help attackers fingerprint the technology stack, allowing them to target known vulnerabilities specific to that framework or version.

POC





Impact

While this issue does not directly compromise the system, it increases the attack surface by exposing framework details. Attackers could use this information for reconnaissance and tailor exploits or scanning tools to the known characteristics of Next.js applications.

Recommendation

Disable the X-Powered-By header in the Next.js configuration to prevent information disclosure.

```
// next.config.js or next.config.ts
const nextConfig = {
  poweredByHeader: false,
};

module.exports = nextConfig;
```

Missing Essential Security Headers

Acknowledged

File Location

kite-foundation-feat-security/next.config.ts
kite_website-feat-security/next.config.js

Description

The current next.config.ts configuration defines a Content-Security-Policy (CSP), but several essential HTTP security headers are either missing or incompletely implemented.

These headers are crucial for protecting against common web-based attacks, ensuring safer browser behavior, and enforcing HTTPS connections.

Specifically, the following security headers are missing or insufficient:

1. X-Content-Type-Options : Prevents MIME-type sniffing.
2. Strict-Transport-Security (HSTS) : Enforces secure HTTPS connections.
3. Content-Security-Policy (CSP) : Present, but missing modern directives like frame-ancestors and unsafe allowances should be minimized.
4. Referrer-Policy : Controls how much referrer data is sent during navigation.

Impact

1. X-Content-Type-Options (Missing): Without this header, browsers may “sniff” content types and interpret non-executable files (like images or text) as executable (e.g., JavaScript), which can lead to cross-site scripting (XSS) or drive-by download attacks.
2. Strict-Transport-Security (Missing): Absence of HSTS allows downgrade attacks where users may unknowingly access the site over HTTP, exposing them to man-in-the-middle (MITM) attacks and session hijacking.
3. Content-Security-Policy (Partially Configured): While a CSP exists, it contains potentially unsafe directives (‘unsafe-inline’, ‘unsafe-eval’) and lacks a frame-ancestors restriction, leaving the application vulnerable to clickjacking and potential script injection if other mitigations fail.
4. Referrer-Policy (Missing): Without a strict referrer policy, sensitive internal URLs or query parameters may be leaked to external domains through the Referer header during navigation.

Recommendation

Implement the following headers for improved security posture.

```
import type { NextConfig } from "next";

const securityHeaders = [
  // Prevent MIME-type sniffing
  {
    key: "X-Content-Type-Options",
    Value: "nosniff",
  },
  // Enforce HTTPS
  {
    key: "Strict-Transport-Security",
    value: "max-age=63072000; includeSubDomains; preload",
  },
  // Control referrer information
  {
    key: "Referrer-Policy",
    value: "strict-origin-when-cross-origin",
  },
  // Strengthen CSP
  {
    key: "Content-Security-Policy",
    value: [
      "default-src 'self' *.gokite.ai;",
      "script-src 'self' 'unsafe-inline' 'unsafe-eval' *.googletagmanager.com
*.gokite.ai;",
      "style-src 'self' data: 'unsafe-inline' *.gokite.ai https://
fonts.googleapis.com;",
      "img-src 'self' data: *.gokite.ai images.pexels.com;",
      "font-src 'self' data: *.gokite.ai https://fonts.gstatic.com;",
      "connect-src 'self' *.gokite.ai https://www.google-analytics.com;",
      "frame-src 'self' *.gokite.ai;",
      "frame-ancestors 'none';",
      "object-src 'none';",
      "base-uri 'self';",
      "form-action 'self' *.gokite.ai;",
      "upgrade-insecure-requests;",
    ].join(" "),
  },
];

const nextConfig: NextConfig = {
  poweredByHeader: false,
  async headers() {
    return [
      {
        source: "/:path*",
        headers: securityHeaders,
      },
    ];
  },
};

export default nextConfig;
```

Outdated Nextjs version in kite_website-feat-security

Acknowledged

Description

The project is currently using an outdated version of Next.js (v14.2.3). Older framework versions may contain known security vulnerabilities, performance issues, or deprecated functionalities.

Frameworks like Next.js frequently release security patches addressing vulnerabilities in SSR (Server-Side Rendering), middleware, and build tooling dependencies (e.g., webpack, SWC, or React integrations).

Impact

Running an outdated framework increases the attack surface and the risk of exploitation of known CVEs in Next.js or its underlying packages.

Recommendation

Upgrade to the latest stable Next.js so vulnerabilities of older versions are mitigated.

React StrictMode Disabled

Acknowledged

File Location

kite-foundation-feat-security/next.config.ts
kite_website-feat-security/next.config.js

Description

The reactStrictMode option is set to false in the Next.js configuration:

```
reactStrictMode: false,
```

This disables React's development-time checks for unsafe lifecycle methods, unexpected side effects, and deprecated patterns. While it has no impact on production, keeping it disabled may reduce visibility of potential issues during development.

Impact

- No security or runtime impact.
- May allow minor coding issues or deprecated practices to remain undetected in development.

Recommendation

Enable React Strict Mode to improve development visibility and maintainability:

```
const nextConfig = {
  reactStrictMode: true,
};
```

NPM Audit Report

Acknowledged

Description

During the course of the audit we ran `npm audit` on the codebase provided. Following is the audit report from the npm audit.

```
# npm audit report

next  0.9.9 - 14.2.31
Severity: critical
Next.js Cache Poisoning - https://github.com/advisories/GHSA-gp8f-8m3g-qvj9
Denial of Service condition in Next.js image optimization - https://github.com/
advisories/GHSA-g77x-44xx-532m
Next.js Allows a Denial of Service (DoS) with Server Actions - https://
github.com/advisories/GHSA-7m27-7ghc-44w9
Information exposure in Next.js dev server due to lack of origin verification -
https://github.com/advisories/GHSA-3h52-269p-cp9r
Next.js Affected by Cache Key Confusion for Image Optimization API Routes -
https://github.com/advisories/GHSA-g5qg-72qw-gw5v
Next.js authorization bypass vulnerability - https://github.com/advisories/
GHSA-7gfc-8cq8-jh5f
Next.js Improper Middleware Redirect Handling Leads to SSRF - https://
github.com/advisories/GHSA-4342-x723-ch2f
Next.js Content Injection Vulnerability for Image Optimization - https://
github.com/advisories/GHSA-xv57-4mr9-wg8v
Next.js Race Condition to Cache Poisoning - https://github.com/advisories/GHSA-
qpjv-v59x-3qc4
Authorization Bypass in Next.js Middleware - https://github.com/advisories/
GHSA-f82v-jwr5-mffw
fix available via `npm audit fix —force`
Will install next@14.2.33, which is outside the stated dependency range
node_modules/next

1 critical severity vulnerability

To address all issues, run:
 npm audit fix —force
```

We would like to mention that most of the above issues stem from the outdated Nextjs version and for each issue certain conditions need to be met.

Recommendation

It is recommended that npm audit fix —force be executed to fix the issues.

Closing Summary

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

Issues of high, low and informational severity were found. The Kite AI team resolved a few and acknowledged the remaining 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

January 2026

For

Kite

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com