






AUDIT REPORT

July 2025

For

nodo

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. Unrestricted Self-Registration Allows Unauthorized Access to Internal Jira Dashboard	11
 Medium Severity Issues	12
1. Hardcoded API Credentials Exposed in Client-Side JavaScript	12
2. HTTP/2 Stream Cancellation Denial-of-Service (CVE-2023-44487) on getprice Subdomain	13
3. Direct Access to Web App Bypassing WAF Protection	14
4. Exposure of Private Keys and Service Accounts	15
5. No Rate-Limiting Enforcement	16
 Low Severity Issues	17
1. Clickjacking	17
2. Deprecated Package in use	18
3. Exposed Kong and Prometheus Metrics Endpoints	19
Closing Summary & Disclaimer	20



Executive Summary

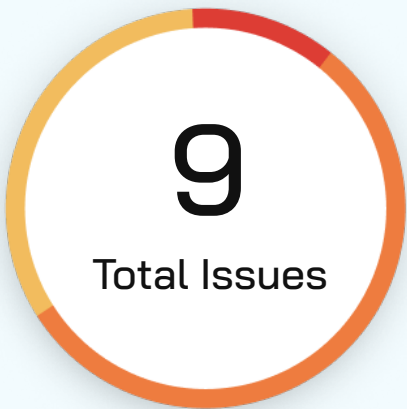
Project Name	Nodo
Protocol Type	Dapp Pentest
Project URL	https://nodo.xyz/
Overview	<p>The Nodo platform is an agentic AI-powered DeFi yield generator and trading ecosystem built primarily on the Sui blockchain, designed to optimize returns for digital asset users through dynamic real-time risk management and autonomous AI agents.</p>
Source Code link	<p>Backend:</p> <ul style="list-style-type: none">- Executor-service: https://git.nodo.xyz/nodo/ai-agent/strategy-execution-service- Data-service: https://git.nodo.xyz/nodo/ai-agent/data-management-service- Frontend code: https://git.nodo.xyz/nodo/frontend/frontend-app- STG site: (https://stg-ai.nodo.xyz/)(https://stg-ai.nodo.xyz/)
Review 1	9th July 2025 - 19th July 2025
Updated Code Received	21st July 2025
Review 2	21st July 2025

Verify the Authenticity of Report on QuillAudits Leaderboard:

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



Number of Issues per Severity



Critical	0 (0%)
High	1 (11.2%)
Medium	5 (55.5%)
Low	3 (33.3%)
Informational	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	5	3	0



Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Unrestricted Self-Registration Allows Unauthorized Access to Internal Jira Dashboard	High	Resolved
2	Hardcoded API Credentials Exposed in Client-Side JavaScript	Medium	Resolved
3	HTTP/2 Stream Cancellation Denial-of-Service (CVE-2023-44487) on getprice Subdomain	Medium	Resolved
4	Exposure of Private Keys and Service Accounts	Medium	Resolved
5	No Rate-Limiting Enforcement	Medium	Resolved
6	Direct Access to Web App Bypassing WAF Protection	Medium	Resolved
7	Clickjacking	Low	Resolved
8	Deprecated Package in use	Low	Resolved
9	Exposed Kong and Prometheus Metrics Endpoints	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

Unrestricted Self-Registration Allows Unauthorized Access to Internal Jira Dashboard

Resolved

Description

The application's Jira instance permits open self-registration, enabling any external user to create an account without validation. Once registered, an attacker can log in and browse internal dashboards, issue trackers, and sensitive project data intended only for employees. This exposes confidential information (e.g. roadmaps, security bugs, personnel assignments) and increases risk of data leakage or further targeted attacks.

Vulnerable URL

<https://jira.nodo.xyz/>

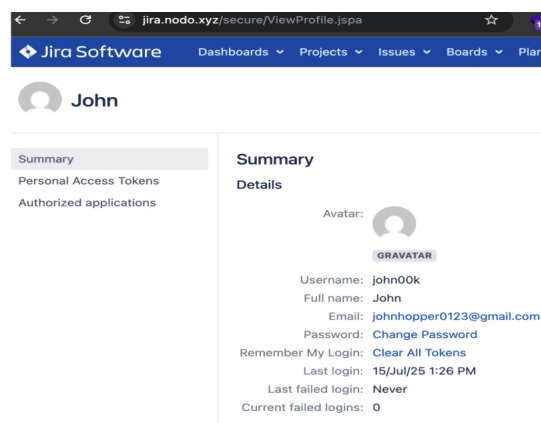
Impact

- **Data Exposure:** Attackers can view application configuration, performance metrics, and potentially sensitive environment variables.
- **Operational Disruption:** Unauthorized users could trigger administrative endpoints (e.g. thread dumps, garbage collection, or shut-down hooks), causing denial of service or degraded performance.
- **Privilege Escalation:** If the dashboard allows configuration changes, attackers may modify log levels, enable debug endpoints, or inject malicious settings.

Recommendation

- **Enable Authentication:** Require strong authentication (e.g., OAuth2, LDAP, or basic auth over HTTPS) for all management endpoints.
- **Audit and Monitor:** Log all accesses to management endpoints and set up alerts for unexpected or repeated requests.

POC



Medium Severity Issues

Hardcoded API Credentials Exposed in Client-Side JavaScript

Resolved

Description

The staging front-end at <https://stg-ai.nodo.xyz> includes sensitive API credentials directly in its bundled JavaScript (index-pQVK5G7o.js). The variables `dZ = "d0b51610b3a2c68205abd8f974fbc87b"`, `fZ = "35416546187b8f78409490e260a1dddc778712c1c46882f787e65a7ab13da9ec"`, `hZ = "https://api-stg.nodo.xyz"` are clearly visible to any user who inspects the source, allowing unauthorized parties to impersonate legitimate clients when calling the backend API.

Vulnerable URL

<https://stg-ai.nodo.xyz/assets/index-pQVK5G7o.js>

Impact

Privilege Escalation: If backend relies solely on these credentials, attackers could access administrative or high-privilege operations.

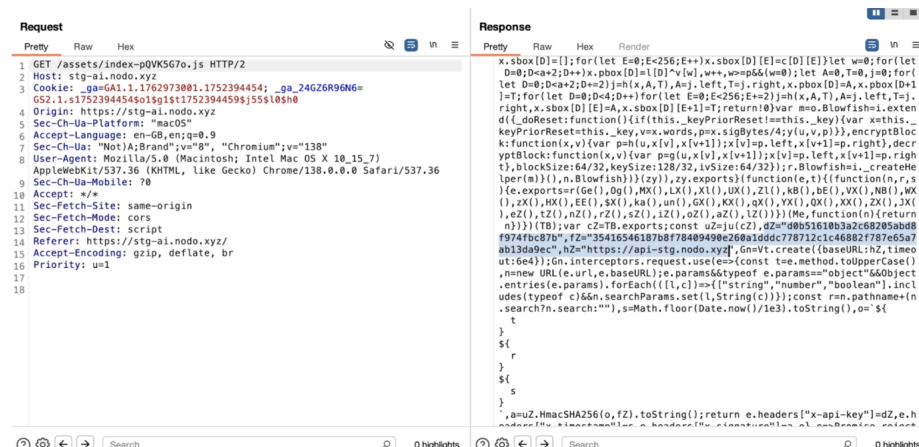
Recommendation

Remove Secrets from Client: Never embed API keys or secrets in client-side code. Move all sensitive calls to a trusted backend.

Use Short-Lived Tokens: Implement an authentication flow (e.g., OAuth2 with JWT) that issues time-limited tokens to the front-end.

Rotate and Revoke Compromised Keys: Immediately rotate the exposed key/secret and revoke any unauthorized tokens

POC



HTTP/2 Stream Cancellation Denial-of-Service (CVE-2023-44487) on getprice Subdomain

Resolved

Description

The getprice service (13.250.151.30) and the main application use HTTP/2, which is vulnerable to a resource-exhaustion attack via rapid stream cancellations. An attacker can open hundreds or thousands of HTTP/2 streams and then send RST_STREAM frames to cancel them repeatedly. Each cancellation forces the server to allocate and free resources, driving up CPU and memory usage and eventually leading to service unavailability.

Vulnerable URL

<https://stg-ai.nodo.xyz/assets/index-pQVK5G7o.js>

Impact

Denial of Service: Exhausts server CPU and memory, causing legitimate requests to be dropped or the server to crash.

Collateral Outage: The DoS against getprice can spill over to shared infrastructure (load balancers, proxies), taking down the main application.

Operational Disruption: Requires manual intervention or restart, impacting availability and potentially resulting in revenue loss and customer dissatisfaction.

Recommendation

Apply Upstream Patch: Upgrade your HTTP/2 library (e.g., nghttp2, httpd, nginx, or your framework) to a version that includes the CVE-2023-44487 fix.

Implement WAF properly to avoid such issues eg: AWS, cloudflare

POC

```
# Example using nghttp2-client to rapidly open and cancel streams
for I in $(seq 1 1000); do
  nghttp -n 1000 -H "User-Agent: DoS-Test" -no-dep https://13.250.151.30/ \
    -method GET \
    -body-ascii "" \
    -no-dep \
    -dump-json | jq .id | xargs -I{} nghttp -n 1 -print-recv -no-dep \
    -method RST_STREAM -body-ascii "{\"id\":{},\"error_code\":0}" https://
13.250.151.30/
done
```



Direct Access to Web App Bypassing WAF Protection

Resolved

Description

The Web Application Firewall (WAF) in front of your services is misconfigured, allowing clients to reach backend applications directly via IP address and specific ports. For example, both 13.250.151.30:443 and 109.123.238.196 on ports 80, 443, 1337, 8000, 8001, 8443, 8444, 9100 respond with valid application content (including GitLab and Jira) without passing through the WAF's filtering or host-header checks. This exposes the raw services—complete with default or sensitive endpoints—that should only be accessible via the public domain names under WAF enforcement.

Impact

Unfiltered Traffic: Attackers can send malicious payloads directly to the apps, bypassing signature-based protection, rate limiting, IP blocks, and other WAF rules.

Credential Brute-Forcing: Login portals reachable without WAF restrictions allow unlimited login attempts, increasing risk of account compromise.

Recommendation

Enforce Host-Header Validation: Configure your WAF to reject requests where the Host header is an IP address or an unrecognized domain.

Restrict Direct IP Access: At the network or load-balancer level, deny incoming traffic to backend server IPs/ports except from the WAF's IPs.

Bind Services to Internal Interfaces: Configure GitLab, Jira, and other apps to listen only on private/internal network interfaces, not on public-facing IPs.

Firewall Rules: Apply security-group or firewall rules to block all ports except those explicitly reverse-proxied through the WAF.

Audit & Monitoring: Continuously scan public IPs for open ports and unexpected services; alert on any new or unauthorized endpoints.

POC

<http://109.123.238.196:9100/>
https://54.225.198.226/users/sign_in
<http://109.123.238.196:8001/>
<https://13.246.146.212/logs/error.log>
<https://13.250.151.30/>

And more



Exposure of Private Keys and Service Accounts

Resolved

Description

Sensitive files such as moonstake-*.json are included in the repository, exposing private keys and Google service account credentials.

Vulnerable File

data-management-service/src/modules/data-access/services/moonstake-384709-f6c4e9853241.json
data-management-service/src/config/app.config.ts - Discloses Doc Pass

Impact

Anyone with repository access can impersonate services, access or modify cloud resources, and decrypt data meant to be secure.

Recommendation

Never commit private keys or service account files to source control.
Store such files in secure vaults and load them at runtime only.

POC

```
shivang@MacBook-Air-1647 data-management-service-main % cat src/modules/data-access/services/moonstake-384709-f6c4e9853241.json
{
  "type": "Service_account",
  "project_id": "moonstake-384709",
  "private_key_id": "f6c4e9853241cd68a7ffdc7671daa887865b22f",
  "private_key": "-----BEGIN PRIVATE KEY-----8nMIEVQIBADANBgkqhkiG9w0BAQFAASCBKcwggSjAgEAAoIBAQCdKqXNDqzwCs0nu4vI1S00MlegUHC5omU/DgaCNzkI2ZRlmo6t74kctfFytr0+mwXJ+88uAx+qwa0nDzTIQDFS/cilaF+809YCI0E/1P17kcd8inhBszFDKbnxbz36pvGEW1j0svP6Gr0nXB/nVh0Y4xi0sn3kLl+J4rTZ4+m3v0NS1JELgIRldcxzPTfg8uHjLhZ988FTfeA9ne7fDY9yyVRL6Nc1Sd4duiRhoxbj50PyES9/th+20c9ixjRldZxcfo8+86KW90H0nc6+7CQCDCicmYvCMsARl1J3z71JUN6nBfukZomnr7AAqr8vumrFth0h3oFNzWxodnaaDkjbVAgMBAAECCgEAAQVZpfa1RjdwA3ymYXb0c0buu772031ymfMBPfhRg0nAvDrwRazvz71vylf8Kxz+BV7phzVodipPx3A3epEsYt9K9dNoNcZgh1Qyrf3o10ntNylR2f1oHfz2190h3Z87/M/zf1mbcutlbgp7ebLWMP0ktQY8K0ePTXm1leCO0bnymfMhETOxsT3alPp11z8mp1kUP4aaq7OyKpvaXKyOei1Q2Lmk7b5Sh7aj10Ap0dnOXUBAYQYTG2LV8kaTONYlnNmH4j00X0dnoW0LAB/Zy8ZMMS0ep0t499/vhnmNtk0nF5uGv7no3Nezt3MhK8CKLM/KR2XbXe3RAwqzCahToQK8gQDulTMubwlyLz0k2fDU0ntrhdLCKKS+x3TFn1pjBXPkPaniB8Jf5xR3Lckgl2scGY8Q8uF0PFFPdzrlXXd00n0jNFkk5L+0NyQJ5F9Hdv0tCwYfWpVnXzXBg1QYbnzfQ0gKNYK+RQe1SQchPy7N5NwzAsfT8CInnd/QcXTGg1J51ABwK8gQDsqon0n2UhhZVhZCv0CadUx8ApMq6SkIX0noRtXocerD0NsnnyYIENxiYtrIXiEukOX6YiXPC3sX8tUfUhoWey/S+e8+Bxkk/0n2LhE9jK7sB5LhycI70SNMYfo8n61cUmpJxU7/mvYQ60f0ez1kw2dKa1Z409P60nHQ1Q7rFWK8gQc0bDM7btgF3R9KIz2/ejNtCBW1s0z2Q7pj5Bh0zk3WAjTvu0M0nbRULqXzReBQ4Wm0CR5+HCwG9sgkKwGWA1/91/GvWTSU1Jqu+d4YrzDnSwjzddP90n7paum1MzHaN6xxF51EzBustAY9Yn0d4LkPORiU0nTbhZR8gE107gELm2wK8gBQ0nKQpuXlqzly/hp01xd0H7gZJ4sEF11ajRBWu010X9POZ3fMMgdJXVGz+eW4P10K0n1jzRxFWaa/ytyVseELuCVvp7630oZJvMukfBZOLgKz2Fy003gQ57B3BhX0xm0nj+BHX8tIfz4drzNvU83rGkC1dPv0j5KMPwPDCP0dAwrf0xehzU20VnJchNHyjTh0nzSe23Rk9b1lyh0IGJQeA51F77wq8FCX8+1v8Uqht1ssKzK2CeBhtyE1z/eWKBfD0nXpZJ/cNaNrotF8mkzpv011TCe0mH+T11pLGLFMU8Yfw73RJurdjXVdWRRKzXZ0nmTwt40gCgchWsjbT0c/gSM=0n-----END PRIVATE KEY-----",
  "client_email": "nodo-40moonstake-384709.iam.gserviceaccount.com",
  "client_id": "117728268182190655629",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/nodo-40moonstake-384709.iam.gserviceaccount.com"
}
```

```
shivang@MacBook-Air-1647 strategy-execution-service-main % cat src/config/app.config.ts
import { registerAs } from '@nestjs/config';
import { IApp } from './interface';

export default registerAs(
  'app',
  () => {
    port: process.env.PORT ? Number(process.env.PORT) : 4000,
    version: '1.0.0',
    name: 'Nodo strategy-execution API',
    description: 'Nodo strategy-execution API API description',
    ratelimit: process.env.RATE_LIMIT ? Number(process.env.RATE_LIMIT) : 100,
    corsOrigin: process.env.CORS_ORIGIN || '',
    enableDoc: process.env.ENABLE_DOC ? Boolean(process.env.ENABLE_DOC) : true,
    docUser: process.env.DOC_USER || 'user_doc',
    docPassword: process.env.DOC_USER || 'Abc@123456',
    languages: ['en', 'vn'],
    defaultLanguage: 'en',
    apiKey: process.env.X_API_KEY,
    applications: ['Nodo'],
    rpc: process.env.RPC_URL,
  },
);
```



No Rate-Limiting Enforcement

Resolved

Description

The service defines a `rateLimit` value in `src/config/app.config.ts` by reading `process.env.RATE_LIMIT` and including it in the `appConfig` object. However, the actual Fastify rate-limiting middleware is never registered in `main.ts` or anywhere else in the codebase. This results in the absence of any real request throttling or abuse prevention logic, leaving the service completely exposed to high-volume request floods.

Vulnerable File

`data-management-service/src/config/app.config.ts` - Defines Rate Limit
`main.ts` - (never registered)

Impact

Without rate limiting, attackers or even accidental user behavior can easily overwhelm the application with excessive requests, leading to denial of service (DoS) conditions. This can impact server resource usage, degrade performance for legitimate users, and potentially take the service offline with minimal effort.

Recommendation

Explicitly register the Fastify rate-limiting plugin in your application's bootstrap/init logic (e.g., inside `main.ts` after Fastify adapter is created):

```
await fastifyAdapter.register(import('@fastify/rate-limit'), {  
  max: appConfig.rateLimit,  
  timeWindow: '1 minute',  
});
```

This will ensure requests are properly throttled and 429 responses are returned after the configured threshold.

Regularly test rate-limiting under load as part of application hardening.



Low Severity Issues

GMX exchange router address might change

Resolved

Description

Clickjacking is a type of attack that tricks a user into clicking on a malicious link or button without their knowledge. This can be done by overlaying an invisible layer over a legitimate link or button on a web page, thus making the user unwittingly click the malicious link or button.

Vulnerable URL

<https://stg-ai.nodo.xyz>

Recommendation

To prevent clickjacking attacks, web developers should use the X-Frame-Options header in their web applications. This header instructs browsers not to render the page in a frame or iframe. Additionally, developers should avoid using legacy code such as ActiveX controls, Flash, and Java Applets as these can be targeted with clickjacking attacks

Impact

If a user is tricked into clicking on a malicious link or button, they may unknowingly give away sensitive information, install malicious software, or be redirected to a malicious website.

POC

<https://clickjacker.io/test?url=https://stg-ai.nodo.xyz>



Deprecated Package in use

Resolved

Description

Multiple JavaScript libraries (multer, fastify, request, axios, tough-cookie, formidable, etc.) are out-of-date and contain high/critical vulnerabilities, including ReDoS, prototype pollution, SSRF, and code execution risks.

Vulnerable File

Package.json, yarn.lock

Impact

Attackers could exploit these flaws to crash the application, execute arbitrary code, bypass validation, or compromise data

Recommendation

Upgrade all vulnerable dependencies to patched versions as indicated by Trivy/YarnAudit. Monitor SCA tools regularly.



Exposed Kong and Prometheus Metrics Endpoints

Resolved

Description

The Kong API gateway and its Prometheus exporter are exposing unauthenticated metrics on publicly accessible endpoints (e.g. /metrics, /kong-metrics). Attackers can directly query these endpoints to retrieve detailed operational data—such as request rates, upstream latency, target health, and internal status codes—without any authentication or IP restriction

Vulnerable File

<http://109.123.238.196:9100/metrics>
<http://109.123.238.196:8001/>

Impact

Information Disclosure: Reveals traffic patterns, API usage, and service health metrics that can aid attackers in planning targeted attacks or performance-based DoS.

Operational Manipulation: An attacker armed with metrics data can identify and stress critical services, then coordinate timed attacks when services are most vulnerable

Recommendation

Network Restriction: Limit access to metrics endpoints to internal networks or specific monitoring IPs via firewall rules or security groups

Monitor Access: Log and alert on any access to metrics endpoints from unexpected sources or outside maintenance windows



Closing Summary

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

Some issues of High, medium, low severity were found. The Nodo team resolved all the issues mentioned in the report

Disclaimer

QuillAudits Dapp audit is not a security warranty, investment advice, or an endorsement of the Nodo. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multi-step process. One audit cannot be considered enough. We recommend that the Nodo Team put in place a bug bounty program to encourage further analysis of the source code by other third parties.



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

July 2025

For

nodo



Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillaudits.com