QuillAudits

# AUDIT REPORT

July 2025

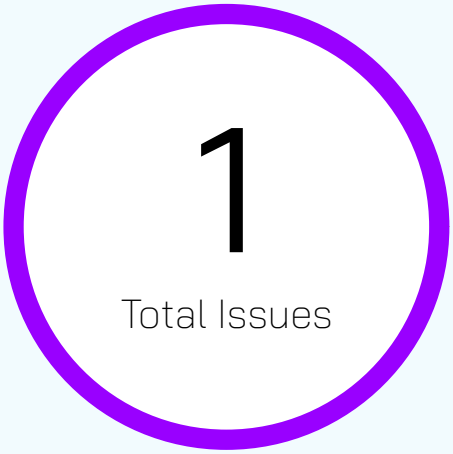
For

nodo

# Executive Summary

| | |
|---|---|
| **Project name** | Nodo |
| **Protocol Type** | AI Agent |
| **Project URL** | https://stg-ai.nodo.xyz/ |
| **Overview** | 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 |
| **Source Code link** | https://github.com/NODO-xyz/AI-Agents |
| **Review 1** | 20th July 2025 - 24th July 2025 |
| **Updated Code Received** | 25th July 2025 |
| **Review 2** | 25th July 2025 |
| **Fixed** | https://github.com/NODO-xyz/AI-Agents/pull/8 |

# Number of Issues per Severity




| | |
|---|---|
| High | 0 (0.00%) |
| Medium | 0 (0.00%) |
| Low | 0 (0.00%) |
| Informational | 1 (100.00%) |

Severity

| Issues | High | Medium | Low | Informational |
|---|---|---|---|---|
| Open | 0 | 0 | 0 | 0 |
| Resolved | 0 | 0 | 0 | **1** |
| Acknowledged | 0 | 0 | 0 | 0 |
| Partially Resolved | 0 | 0 | 0 | 0 |

# Checked Vulnerabilities

- ☑ Environment Variable Validation
- ☑ Adversarial Prompt Injection
- ☑ Dangerous Function Usage (eval
- ☑ exec)
- ☑ Unsafe Deserialization (pickle
- ☑ yaml.load)
- ☑ Server-Side Request Forgery (SSRF)
- ☑ Arbitrary File Write
- ☑ Hardcoded Secrets/API Keys
- ☑ Insecure Randomness
- ☑ LLM Response Execution
- ☑ Dynamic getattr() or Module Dispatch
- ☑ Insecure Subprocess Execution
- ☑ Directory Traversal (via open/write)
- ☑ Input Validation & Output Encoding
- ☑ Authentication & Authorization Controls
- ☑ Logging Sensitive Data
- ☑ Race Conditions on File Access
- ☑ Tamperable Model Configs
- ☑ Misconfigured AI Role or System Prompts
- ☑ Denial of Service via Missing Timeouts
- ☑ Safe Use of External Libraries
- ☑ Model Misalignment Triggers (LLM Safety)
- ☑ And more.

# Techniques & Methods

Throughout the pentest of applications, 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 | Nabbu | Dirbuster |
| DNSEnum | Turbo Intruder | SQLMap |
| Acunetix | Nmap | Horusec |
| Neucil | Metasploit | Postman |
| Netcat | Nessus | and many more... |

# Types of Severity

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

### High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

### Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

### Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

### Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

# Types of Issues

**Open**

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

**Resolved**

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

**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.

# Informational Severity Issues

## Environment Variable Validation

Resolved

### Description

The application uses dotenv.load_dotenv() to load environment variables (e.g., OPENAI_API_KEY, DB_URL, SECRET_KEY) but does not verify their presence or correctness at runtime. Across multiple files (e.g., agent/llm/__init__.py, src/settings.py, flowx/utils.py), these variables are accessed directly. This access pattern assumes the variable exists and is valid. If the .env file is missing, incomplete, or misconfigured, the application will raise a KeyError, potentially crash, or fall back to insecure defaults without logging or alerting.

### Vulnerable File

agent/llm/__init__.py
src/settings.py
backtest/, flowx/, trading/ utils/services

### Impact

If a required .env variable is missing or malformed:
The application crashes unexpectedly → DoS
The application may silently misbehave (e.g., empty credentials)
No fallback logic or exception handling
Security policies assumed in env (e.g., sandbox, proxy, API key scopes) may not be enforced

### Recommendations

Create a config.py:
import os

REQUIRED_ENV_VARS = ["OPENAI_API_KEY", "DB_URL", "SECRET_KEY"]
missing = [var for var in REQUIRED_ENV_VARS if not os.getenv(var)]

if missing:
 raise RuntimeError(f"Missing required environment variables: {missing}")

Then import config.py early in your project entrypoint.

# Closing Summary

In this report, we have considered the security of the Nodo. We performed our audit according to the procedure described above. Informational severity was found. The Nodo team resolved the mentioned issue

# 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 recom-mend 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

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com