QuillAudits

AUDIT REPORT

October 2025

For

Naga

Table of Content

Executive Summary

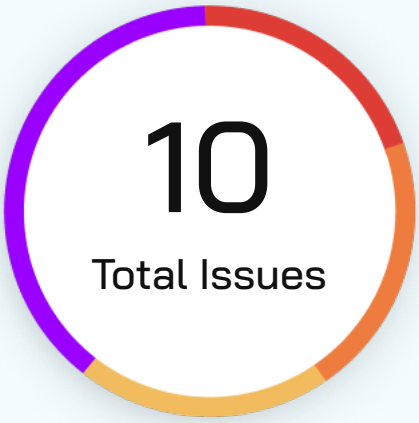
Project Name	Naga
Protocol Type	DEX, Router
Project URL	https://naga.fi/
Overview	This protocol allows swaps between cryptocurrencies with high-end price protection, oracle fallback and multi-leg routing using an off-chain quoter that compares prices to Chainlink's on-chain oracle.
Audit Scope	The scope of this Audit was to analyze the Naga Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/naga-fi/naga-audit
Branch	main
Contracts in Scope	forex2/Forex2Router01.sol forex2/Forex2AggregatorRouter01.sol forex2/Forex2Vault.sol forex2/Forex2VaultFactory.sol oracle/TakeoverableOracle.sol oracle/TimedChainlinkOracle.sol
Commit Hash	f9947fe051cfae30ffc4d0b68e5f045e9855fd36
Language	Solidity
Blockchain	Arbitrum, Unichain
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	8th October 2025 - 13th October 2025
Updated Code Received	19th October 2025 - 13th October 2025
Review 2	22nd October 2025 - 23rd October 2025

Fixed In	Fixes branch: c1bdea01aece25f9dc5449ea40a26053885c4e41 **Tree hash:** **42d874b5114e8a71ea4976649525c352bad48e59879487bf6cff66908600dfb3**

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity



Critical	0 (0%)
High	2 (20%)
Medium	2 (20%)
Low	2 (20%)
Informational	4 (40%)

Severity

Issues	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	1	2	1	3
Partially Resolved	0	0	0	0	0
Resolved	0	1	0	1	1

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Snapshot/bound bucket mismatch disables the clamp	High	Acknowledged
2	Infinite approvals granted cannot be revoked	High	Resolved
3	Unvalidated vaults + externally-initializable vaults enable non-official vault misuse	Medium	Acknowledged
4	Missing amount and staleness check in TakeoverableOracle	Medium	Acknowledged
5	LPs overpay on swaps because protocolFees are not taken from the gross totalAmountIn	Low	Resolved
6	Zero–answer snapshot makes slot “re–open” and turns off bounds	Low	Acknowledged
7	AmountOutMismatch error parameter order inconsistency	Informational	Resolved
8	Unbounded maxChangePerDayE18 parameter	Informational	Acknowledged
9	balanceOf(address(this)vs computed swapAmount	Informational	Acknowledged
10	To note, quality of life improvements	Informational	Acknowledged

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
- 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.

High Severity Issues

Snapshot/bound bucket mismatch disables the clamp

Acknowledged

Path

TakeoverableOracle.sol

Function Name

`snapshot(), _boundAnswerToSnapshotIfAny()`

Description

snapshot() stores answers keyed by a 6-hour bucket (00:00, 06:00, 12:00, 18:00) while _boundAnswerToSnapshotIfAny() looks up answers keyed by a 24-hour (midnight) bucket. The write and read keys rarely coincide, so answerSnapshots[ts] reads 0 which makes the if (lastDayAnswer > 0) branch to be skipped, and the raw oracle answer is returned without clamping. This fully bypasses the intended maxChangePerDayE18 rate-limit even when snapshots are taken "today."

Impact

The price bound is effectively off allowing unbounded feed moves pass through.

Likelihood

No special permissions needed for this to occur.

POC

1. Time = 10:37 UTC; keeper (or any EOA) calls snapshot():
2. Writes answerSnapshots[floor(10:37/6h)*6h] = answer → key = 06:00.
3. A consumer later calls latestRoundData(): _boundAnswerToSnapshotIfAny() computes daily key = floor(10:37/24h)*24h = 00:00.
4. lastDayAnswer = answerSnapshots[00:00] == 0 → bound not applied → returns raw answer.
5. Even if additional snapshots are taken at 12:00 or 18:00, the read still uses 00:00, so the clamp never activates that day.

Recommendation

Use the same bucket in both functions (either both 6–hour or both daily). Optional: if no snapshot exists for the bucket, revert or fall back to the most recent prior snapshot depending on availability vs. safety trade–offs.

Naga Team's Comment

We will have a cronjob that makes snapshots automatically. There is a 6-hour window from 00:00 to make a new snapshot. If no new snapshots, fallback to trusting the oracle. We don't expect to miss any snapshots. The subtraction is there to allow new-day snapshot to be available.

Auditor's Comment

If expecting only the midnight snapshot to be the source of truth, the comment says you'd try to fallback to the most recent snapshot in case the current one fails to provide an answer. The current implementation effectively skips 1 snapshot and goes 2 snapshots behind to make any reads.

Naga Team's Comment

Acknowledged

Infinite approvals granted cannot be revoked

Resolved

Path

Forex2AggregatorRouter01.sol

Function Name

`_safeCallZeroX`

Description

The isSettlerCache is never updated even if the addresses get compromised, this could pose a challenge because these addresses receive infinite approvals in _safeCallZeroX(). The attack surface can be reduced by having a setter function to reset this privilege. In the case of signer compromise or an address not meeting the requirements described below, they would be able to manipulate token balances present in the Router (if any).

```
zeroXQuote.to == allowanceHolder ||
zeroXQuote.to == settlerRegistry.ownerOf(2) ||
zeroXQuote.to == settlerRegistry.prev(2);
```

Recommendation

Make it possible to revoke infinite approvals to privileged addresses.

Medium Severity Issues

Unvalidated vaults + externally-initializable vaults enable non-official vault misuse

Acknowledged

Path

Forex2Router01.sol, Forex2Vault.sol

Function Name

`Forex2Router01.swap, Forex2Vault.initialize`

Description

The router accepts vault addresses from the signed quote and doesn't check they were created by the official factory. The vault contract also exposes initialize as an external function (guarded only by an initialized flag), allowing independently deployed vaults to be first-called and controlled by anyone. Combined, a misused/compromised QUOTER_ROLE can direct swaps through attacker-controlled "vaults" that implement the expected pull interface but aren't "official," bypassing the intended allowlist/factory control.

Impact

Swaps can route user funds to non-official vaults.

Recommendation

Enforce an on-chain allowlist in Forex2Router01.swap, require factory.isVault(quote.parts[I].vault) for each part

For Forex2Vault, remove external initialize by:

1. Making initialization constructor-based; or
2. Adding an immutable factory and an onlyFactory initialize; or
3. Keeping initialize but restricting it with onlyFactory and a one-time guard.

Naga Team's Comment

Acknowledged, we assume the quoter provides the correct vaults

Missing amount and staleness check in TakeoverableOracle

Acknowledged

Path

TakeoverableOracle.sol

Function Name

`getRoundData()`

Description

The oracle does not check for non-positive/zero answers or stale data (older than maxStaleness) before returning them, unlike the referenced Chainlink oracle which reverts on such conditions.

Impact

Oracle reads may return stale or invalid data.

Recommendation

Enforce validity and freshness inside getRoundData before returning.

Naga Team's Comment

Acknowledged, No check needed as this is a proxy contract

Low Severity Issues

LPs overpay on swaps because protocolFees are not taken from the gross totalAmountIn

Resolved

Path

contracts/forex2/Forex2Router01.sol

Path

`swap(), getAmountOutAt()`

Description

The router calculates the oracle cap using the gross input totalAmountIn, but transfers only net input to the vault (after deducting protocolFeePctE18). Because getAmountOutAt is monotonic in amountIn, cap(gross) > cap(net), allowing quotes up to the inflated cap while the vault receives less value. The difference is paid by LPs, effectively siphoning more tokens from them than should be paid (up to the difference between protocolFeePctE18 and lpFeePctE18.

Impact

Economic loss to LPs proportional to protocol fee and trade size; accumulates with volume because LPs pay out more output than a net based cap.

Likelihood

Guaranteed to occur in every swap that has quote.parts.length >= 1.

POC

Assumptions:

- Price: 1 USDC → 4 MYRC (before LP fee)
- lpFee = 0.2% (0.002)
- protocolFee = 0.15% (0.0015)

Calculations:
Gross-output cap (uses totalAmountIn = 10,000):
- Pre-fee out = 10,000 × 4 = 40,000 MYRC
- After LP fee = 40,000 × (1 − 0.002) = 39,920 MYRC

Net sent to vault (after protocol fee):
- Protocol fee in USDC = 10,000 × 0.0015 = 15
- Net input = 10,000 − 15 = 9,985 USDC

Net-based cap (what it would be if capped on net):
- Pre-fee out = 9,985 × 4 = 39,940 MYRC
- After LP fee = 39,940 × 0.998 = 39,860.12 MYRC

Quick formula

Excess = grossIn × price × (1 – lpFee) × protocolFee
= 10,000 × 4 × 0.998 × 0.0015 = 59.88 MYRC overpaid on swap.

Recommendation

1. Compute totalAmountInNet = Σ(amountIn - protocolFee) and use that for getAmountOutAt;
2. Transfer grossAmount to the vault after getting the sum from all parts of the quote and have the vault pay protocol fees out based on the gross amount transferred.

Naga Team's Comment

This is by design. The protocol fee is taken from LPs. This is similar to how Uniswap V2 takes the fees from LPs by minting shares to feeTo. Here we just take the fee directly in tokens.

Auditor's Comment

The severity of this issue has been downgraded because it does not propose a direct threat to the LPs as these fees go directly to them.

Zero—answer snapshot makes slot “re—open” and turns off bounds

Acknowledged

Path

TakeoverableOracle.sol

Path

`getRoundData()`

Description

snapshot() allows answer == 0 and stores 0. The guard requires answerSnapshots[ts] == 0 to snapshot and, with stored 0, it keeps allowing repeats. In _boundAnswerToSnapshotIfAny, bounding triggers only if lastDayAnswer > 0, so 0 disables clamping.

Impact

If the answer is not clamped, any values returned from the oracle call will be used.

Recommendation

Do not store 0 as a value for the answerSnapshots mapping.

Naga Team’s Comment

Acknowledged, if the price is zero, there should be no clamping

Informational Issues

AmountOutMismatch error parameter order inconsistency

Resolved

Path
Forex2Router01.sol

Function Name
`AmountOutMismatch(uint256 cap, uint256 actual)`

Description
The custom error AmountOutMismatch(uint256 cap, uint256 actual) defines parameters as (cap, actual) but the require passes (actual, cap), leading to confusing revert diagnostics and tooling mismatches.

Recommendation
Emit (cap, actual) instead of (actual, cap).

Unbounded maxChangePerDayE18 parameter

Acknowledged

Path
TakeoverableOracle.sol

Function Name
`setMaxChangePerDay()`

Description
Admin can set maxChangePerDayE18 to arbitrarily high values, which can effectively disable snapshot–based bounding. No cap or sanity check is enforced.

Recommendation
Add a sanity check for the maxChange parameter.

Naga Team's Comment
Acknowledged, we assume the admin sets a reasonable value

balanceOf(address(this)vs computed swapAmount

Acknowledged

Path

Forex2AggregatorRouter01.sol

Path

`swapWithFirstLeg(), swapWithLastLeg()`

Description

Using the amount deltas (difference between initial amount and amount after swaps) will yield more precise transfer values for token swaps than transferring the router contract balance.

Naga Team's Comment

Acknowledged

To note, quality of life improvements

Acknowledged

Description

1. The require statement checks that the `recovered` address has QUOTER_ROLE at execution time. Note: This only supports EOAs. Contract wallets would need EIP-1271 which isn't implemented here.
2. The offchain quoter service is ALWAYS trusted to provide honest quotes and vaults used for swaps
3. The ThirdPartyOracle is an immutable value with a check that will fail when switching to a new oracle – this means the oracle is expected to never update decimals.
4. Unused imports exist because the Euler Price lib (ChainlinkOracle) already imports some of the libraries/contracts internally.

Naga Team's Comment

Acknowledged

1. Only EOA is fine
2. This is by design
3. Price feed decimals are assumed to never change
4. Acknowledged

Functional Tests

Some of the tests performed are mentioned below:

- ✓ Should swap with a single vault
- ✓ Should swap with multiple vaults
- ✓ Should swap MYRC to USDC
- ✓ Should swap USDC to MYRC
- ✓ Should swap tokens via AggregatorRouter
- ✓ Should swap via aggregatorRouter
- ✓ Should send excess tokens from swaps to excessRecipient
- ✗ Fallback should hit the closest snapshot before going to the oracle fallback
- ✓ Should allow only the quoter to provide valid quotes
- ✓ Should check for oracle data staleness
- ✓ Should validate answers

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
Forex2Router01	swap, getAmountOut, getAmountOutAt, setProtocolFeePctE18, setLpFeePctE18	Overpayment to taker due to cap mis-accounting if cap is computed on gross but vault receives net (LP fee deducted) Missing isVault check allows interacting with arbitrary user-provided “vault” addresses Signature replay or spoofing Oracle mismatch/round manipulation: Fee-griefing
Forex2Vault	initialize, execute, pull	initialize externally callable execute arbitrary call by admin pull by router only ERC20 interactions
Forex2VaultFactory	createVault	initialize externally callable execute arbitrary call by admin pull by router only ERC20 interactions

Contract	Function	Threats
TimedChainlinkOracle	getQuoteWithRoundId, getQuoteAt	Staleness or negative/zero answers Decimals scaling mismatch
TakeoverableOracle	latestRoundData, getRoundData, takeover, snapshot, setMaxChangePerDayE18	upstream feeds Bounding mismatch buckets (daily vs snapshot cadence): Owner misuse

Closing Summary

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

2 High, 2 Medium, 2 Low and 4 Informational Severity issues were found. These issues have been resolved, noted and acknowledged by the Naga team.

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

October 2025

For

Naga

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com