SMART CONTRACTS REVIEW

zokyo

June 13th 2025 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

ZOKYO AUDIT SCORING LAYERK

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: 0 points

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 1 Medium issue: 1 resolved = 0 points deducted
- 1 Low issue: 1 resolved = 0 points deducted
- 3 Informational issues: 2 resolved and 1 acknowledged = 0 points deducted

Thus, the score is 100

TECHNICAL SUMMARY

This document outlines the overall security of the LayerK smart contract/s evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the LayerK smart contract/s codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the LayerK team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the LayerK repository:
Repo: zip file
SHA: 916988011c34f3f476bbbf0c7d02b82f1a8603d0e9eec92e1dd14aaef950062b
Last SHA: 676e5e7f37b98bacb0123a1f8c91a3832f223ec5

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- NodeK.sol
- GLYK.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of LayerK smart contract/s. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contract/s by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

LayerK is a technology company focused on hardware and software innovations that leverages advanced computing and blockchain technology to empower individuals toward achieving independence. The company's vision centers on democratizing resource access and creating a decentralized, sustainable, and fair global economy where everyone can participate in the Web3 era.

The LayerK ecosystem encompasses multiple components including the Chain, a specialized blockchain with EVM compatibility and integrated bridging capabilities, and a comprehensive staking infrastructure. Zokyo was tasked with conducting a security review of two critical smart contracts within this ecosystem: the GLYK governance token contract and the NodeK staking contract.

The GLYK token implements a unique design pattern where transfers are completely disabled, serving exclusively as a governance token that can only be minted and burned by authorized contracts. The contract follows the UUPS upgradeable proxy pattern and incorporates access controls to restrict minting and burning operations to specifically authorized addresses, ensuring controlled token supply management.

The NodeK contract serves as a core staking mechanism for the LayerK ecosystem, implementing a sophisticated reward distribution system. The contract utilizes EIP-712 signature validation for member authorization, supports external wallet management for governance token distribution, and maintains comprehensive stake tracking with business ID associations. The system allows users to stake native tokens and receive GLYK governance tokens as rewards, creating an integrated economic model between the platform's utility and governance layers.

Overall the code is well written, well commented and relies on battle hardened libraries which have withstood the test of time. The findings discovered during the audit ranged between Medium and Informational due to the strong adherence to security, the simplicity of the code and reliance of trusted libraries. Most of the findings addressed niche edge case scenarios should certain cases arise and their impacts. Zokyo has created recommendations which will resolve these issues.

Following the audit, a fix review process should be implemented to verify that the recommended changes properly address the identified vulnerabilities without introducing new security risks. Zokyo wishes the team at LayerK all the very best when moving to deploy to a production environment.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the LayerK team and the LayerK team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Governance Token Address Change Compromises Accounting Integrity	Medium	Resolved
2	Inconsistent Access Control Implementation	Low	Resolved
3	Duplicate zero address check for external wallet	Informational	Resolved
4	Redundant storage reads	Informational	Resolved
5	Floating pragma	Informational	Acknowledged

Governance Token Address Change Compromises Accounting Integrity

Location: NodeK.sol

The `setGovernanceToken` function in NodeK.sol allows the owner to arbitrarily change the address of the governance token (glykToken) without: ensuring backward compatibility with previous accounting and providing migration logic for existing token balances.

This can invalidate all prior accounting (e.g., rewards, balances, or voting power) tied to the previous token address, as the contract will now reference a new token contract.

Impact:
And immediate risk is that users' accrued rewards in the old token become worthless.

Recommendation:
Enable users to claim from the old token within a migration grace period.

Inconsistent Access Control Implementation

Location: NodeK.sol

The contract inherits both `Ownable` and `AccessControl`, leading to redundant ownership checks as:

- `Ownable` provides a single owner via onlyOwner modifier.
- While `AccessControl` allows fine-grained roles (e.g., DEFAULT_ADMIN_ROLE, MINTER, BURNER, ...etc).

There is also `admin` address set by `setAdmin` function which is unnecessary if `AccessControl` is already used for role management.

Impact:

- Increased attack surface as Multiple admin-setting functions increase risk of misconfiguration.
- Confusion in permission management (e.g., owner vs. DEFAULT_ADMIN_ROLE).

Recommendation:

Remove `Ownable` and `admin/setAdmin` dependency as this can be replaced by new roles from `AccessControl` for better consistency.

Example:

```
// Remove:
function setAdmin(address _admin) external onlyOwner { ... }

// Use instead:
function setAdmin(address _admin) external onlyRole(DEFAULT_ADMIN_ROLE)
{
    _grantRole(SOME_NEW_ADMIN_ROLE, _admin);
}
```

INFORMATIONAL-1 | RESOLVED

Duplicate zero address check for external wallet

Location: NodeK.sol

The _distributeGovernanceTokens() function has a check to ensure that the user's external wallet is not zero address.
However, the _distributeGovernanceTokens() function is an internal function and it's called in the approveExternalWallet(), updateReward(), and _createStakeCore() functions, and these functions already have the same checks.

Impact:
Repeating the same check unnecessarily increases the gas consumption of the function, making it less efficient.

Recommendation:
Remove the zero address check for the external wallet in the _distributeGovernanceTokens() function.

INFORMATIONAL-2 | RESOLVED

Redundant storage reads

Location: NodeK.sol

The unstake() function reads rec.stakeAmount value from storage multiple times. Reading from storage is significantly more expensive in terms of gas costs compared to reading from memory.

Impact:
Redundant storage reads increase the gas consumption of the function, making it more expensive for users to interact with the contract.

Recommendation:
Cache rec.stakeAmount value in memory.

Floating pragma

Location: GYLK.sol, NodeK.sol

The contracts use a floating pragma version `(^0.8.22)`. Contracts should be deployed using the same compiler version and settings as were used during development and testing. Locking the pragma version helps ensure that contracts are not inadvertently deployed with a different compiler version.

Impact:
Different compiler versions may introduce changes in behavior, leading to unexpected outcomes when the contract is deployed or interacted with.

Recommendation:
Consider locking the pragma version to a specific, tested version to ensure consistent compilation and behavior of the smart contract.

Client comment: We acknowledge this and proceed with the same.

	NodeK.sol GLYK.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

We are grateful for the opportunity to work with the LayerK team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the LayerK team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo