QuillAudits

AUDIT REPORT

February, 2025

For

Zynk Labs

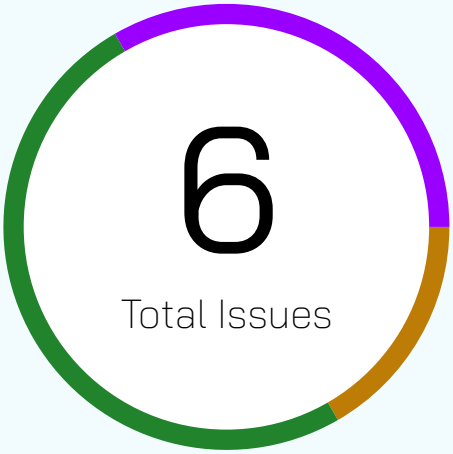
Table of Content

Executive Summary

Project name	ZynkLabs
Overview	ZynkProtocol is a lending system where tokens are lent to verified partners who repay with fees. Each loan is tracked with a unique orderId, and repayments require validator signatures with a validity timestamp to prevent expired transactions.
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Blockchain	Arbitrum
Audit Scope	The scope of this Audit was to analyze the ZynkLabs Smart Contracts for quality, security, and correctness. https://github.com/MasterAtWork/Zynk-Protocol-Smart-Contracts
Contracts In Scope	1. contracts/ZynkProtocol.sol 2. contracts/ZynkTokenManager.sol 3.contracts/ZynkWhitelistManager.sol
Fixed In	NA
Commit Hash	Main: e415083016c04c8d2fce359b574ac720612c5f63
Language	Solidity
Method	Manual Review,Functional testing and Automated Scan
Review 1	20th February 2025 - 3rd March 2025

Updated Code Received	NA
Review 2	NA

Number of Issues per Severity



High	0 (0.00%)
Medium	1 (16.67%)
Low	3 (50.00%)
Informational	2 (33.33%)

Severity

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

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

- [x] Using inline assembly
- [x] Unsafe type inference
- [x] Style guide violation
- [x] 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.
- Implementation of ERC standards.
- 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 statistical analysis.

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.

Medium Severity Issues

Use abi.encode instead of abi.encodePacked

Acknowledged

Path

ZynkWhitelistManager.sol

Function

checkSum()

Description

The function checkSum currently uses abi.encodePacked to hash the partnerId. While abi.encode-Packed creates a compact encoding, it can lead to potential collisions when concatenating multiple dynamic types. Using abi.encode ensures a more structured encoding, reducing ambiguity and preventing unintended hash collisions.

Recommendation

Replace abi.encodePacked(partnerId) with abi.encode(partnerId) in the checkSum function to ensure a more structured encoding.

Teams Comment

ZYNKLabs Team's Comment : abi.encodedPacked uses less gas compared to abi.encode and it's purely to the engineer on how to manage the data inside. Nonce will be incremented at each send tx, probability is close to zero to have a collision

Low Severity Issues

Centralization Risk

Acknowledged

Description

Even though all the actors in the protocols are trusted, it is still important to remember that the contract has many points of failure in case of compromised or trusted actors like admin, partners, manager etc.

Recommendation

Ensure that admin roles are multisig and trusted roles are not malicious

Use ownable2Step instead of Ownable

Acknowledged

Description

The contract currently uses direct ownership transfer mechanisms. Using Ownable2Step ensures a safer transfer of ownership by requiring the new owner to accept ownership explicitly before it is finalized. This prevents accidental loss of ownership.

Recommendation

Replace the current ownership mechanism with OpenZeppelin's Ownable2Step, which provides a more secure two-step ownership transfer process

Teams Comment

ZYNKLabs Team's Comment : Will consider in upcoming version.

Hardcoded chainId might be problematic

Acknowledged

Path

ZynkProtocol.sol

Description

The contract initializes chainId using block.chainid in the constructor, but it does not account for potential chain migrations or forks where chainId may change. This can cause issues with cross-chain validation and future compatibility.

Recommendation

Instead of hardcoding chainId in the constructor, dynamically retrieve block.chainid in functions that require it or allow for an updatable chain ID via an admin-controlled function.

Teams Comment

ZYNKLabs Team's Comment : Not all the networks support block.chainId and also all the versions of solidity, I don't see this as an issue.

Informational Severity Issues

Change confusing function name

Acknowledged

Path

ZynkProtocol.sol

Description

The function revokeReplenish() marks an order as active instead of revoking it, which is counterintuitive and can lead to misunderstandings during contract interactions.

Recommendation

Rename revokeReplenish() to something more intuitive, such as markOrderActive() or reactivateOrder(), to better reflect its actual functionality.

Teams Comment

ZYNKLabs Team's Comment :My intention is that we are revoking the replenish that is already done, hence wrote the function name as revokeReplenish.

Use solmate's safeTransferHelper instead

Acknowledged

Path

ZynkProtocol.sol

Description

The contract currently uses a custom TransferHelper library for token transfers. Solmate's SafeTransferLib is a more optimized and widely used alternative, providing better gas efficiency and security.

Recommendation

Replace TransferHelper with SafeTransferLib from Solmate to improve gas efficiency and security in token transfers.

Teams Comment

ZYNKLabs Team's Comment : TransferHelper library is used by Uniswap, Sushiswap and many other protocols, I don't see this as an issue at the moment, we can consider upgrading in upcoming versions.

Functional Tests

Some of the tests performed are mentioned below:

- [PASS] ZynkTokenManager.sol contract adds and removes token correctly with a managing access control
- [PASS] ZynkWhitelistManager.sol contract adds and removes whitelisted operational partners' wallets with correctly managing access control.
- [PASS] ZynkWhitelistManager.sol contract adds and removes whitelisted Deposit partners' wallets with correctly managing access control.
- [PASS] The manager manages updateTokenManager updateWhitelistManager revokeReplenish and pause and unpause mechanism.
- [PASS] send function transfers amount to partner correctly and update nonce and store the data correctly.
- [PASS] replenish the function transfer funds to the payback address as per expectation.

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.

Closing Summary

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

Some issues of low,Informational and medium severity were found. In the End,Zynk Labs Team Acknowledged all 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 secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

February, 2025

For

Zynk Labs

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com