GA GUARDIAN

Norlend

Lending Contracts

Security Assessment

February 24th, 2025

Summary

Audit Firm Guardian

Prepared By Robert Reigada, Daniel Gelfand, Zdravko Hristov,

Osman Ozdemir, Mark Jonathas, Michael Lett

Client Firm Norlend

Final Report Date February 24, 2025

Audit Summary

Norlend engaged Guardian to perform a follow-up security review of its borrowing lending platform built on top of SparkLend. From the 4th of February to the 7th of February, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement only Medium and Low issues were uncovered and promptly remediated by the Norlend team which indicates an overall healthy design and implementation. Following their remediation Guardian believes the protocol to uphold the functionality described for the Lending product.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Ethereum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/vanir-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	Norlend
Language	Solidity
Codebase	https://github.com/MEWB-AS/vanir-contracts
Commit(s)	Initial commit: ec87b0d5801d4b74fa6a1fbb6e8c8d29ebabcc9d Final commit: 0a0e05b49e654f944f15fd07b3f996c771aecdd2

Audit Summary

Delivery Date	February 24, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	3	0	0	2	0	1
Low	6	0	0	3	0	3

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	User Can Pass Arbitrary Fee Amount	Logical Error	Medium	Acknowledged
M-02	Failed Approvals With USDT	Logical Error	Medium	Resolved
M-03	Lost Borrow Fee	Validation	Medium	Acknowledged
L-01	LendingPoolAddress Points To Registry Instead	Deployment	Low	Resolved
L-02	No Controls On Liquidation Fee Amount	Validation	Low	Resolved
L-03	Failed Liquidations With Low Target LTV	Warning	Low	Resolved
L-04	Failing Tests	Warning	Low	Acknowledged
L-05	No Events Emitted On Funds Rescue	Event	Low	Acknowledged
L-06	Frozen Pool Will Lead To Fail Liquidations	Warning	Low	Acknowledged

M-01 | User Can Pass Arbitrary Fee Amount

Category	Severity	Location	Status
Logical Error	Medium	Vanir.sol	Acknowledged

Description

Function `borrow` accepts an arbitrary `feeAmount` through a `loan` call. If Norlend expects a fee to be paid on borrow, a user can just directly interact with the contract and pass a `feeAmount` of zero to preserve as much capital.

Recommendation

Add validation on the expected `feeAmount` for Norlend. Alternatively, consider tieing feeAmount to a fixed percentage of the borrowed/liquidated amount rather than letting the caller supply an arbitrary fee, implement a fixed percentage model (e.g., `feeAmount = (borrowedAmount * feePercentage) / 1e18`).

Resolution

Norlend Team: We own all the wallets our customers utilize, so we are not concerned with external parties calling the contract using their own wallets.

M-02 | Failed Approvals With USDT

Category	Severity	Location	Status
Logical Error	Medium	Vanir.sol	Resolved

Description

Function `repay` approves `amount` and then calls `SparkLend` to repay the amount. The issue is that the `paybackAmount` within the BorrowLogic is not necessarily equal to `amount` passed, so a portion of the approved amount will be not be utilized.

This will lead to DoS with tokens such as USDT which require a 0 approval initially.

Recommendation

For all allowances in Norlend, use `SafeERC20`'s `forceApprove` which will force the allowance to go to zero initially to handle tokens such as USDT. Also, use it after an external call to set the approval to zero after the approval is no longer necessary.

Resolution

Norlend Team: Added a `forceApproval` function to the contract that performs the functionality as needed. We also reset the allowance where needed.

Guardian Team: After function `swap` is performed, the SwapRouter may still have some token approval leftover if the entire `amountIn` was not utilized.

M-03 | Lost Borrow Fee

Category	Severity	Location	Status
Validation	Medium	Vanir.sol	Acknowledged

Description

When users borrow via Norlend, a part of the borrowed assets is paid as a fee to the `feeAddress`, which is an arbitrary passed parameter, but is validated to be a whitelisted address.

However, whitelisted addresses are also the swap routers and the lending pools. This means users may choose to pay the fee to any other whitelisted address resulting in loss of funds for Norlend.

Recommendation

Create a separate validation for the `feeAddress`.

Resolution

Norlend Team: If an external party wishes to utilize the contract without disrupting its function, they are free to do so.

L-01 | LendingPoolAddress Points To Registry Instead

Category	Severity	Location	Status
Deployment	Low	DeployVanir.s.sol	Resolved

Description

The deployment script currently passes `0x02C3eA4e34C0cBd694D2adFa2c690EECbC1793eE` (the `PoolAddressesProvider` of Spark) to the `Vanir` constructor, instead of passing `0xC13e21B648A5Ee794902342038FF3aDAB66BE987` (the Spark `LendingPool` address on Ethereum mainnet).

As a result, the Vanir contract is initialized with the wrong contract reference thereby preventing it from interacting correctly with the Spark lending protocol.

If the contract is meant to integrate with Spark Lend on mainnet, the `LendingPool` address (`0xC13e21B648A5Ee794902342038FF3aDAB66BE987`) should replace the `PoolAddressesProvider` address in the script.

Using the `PoolAddressesProvider` directly is incorrect in this context because Norlend specifically needs the active `LendingPool` to perform supply/borrow/repay operations, not just an address provider.

Recommendation

Update the constructor call in `DeployVanir` (or any relevant deployment script) to use `0xC13e21B648A5Ee794902342038FF3aDAB66BE987` instead of `0x02C3eA4e34C0cBd694D2adFa2c690EECbC1793eE`.

Resolution

Norlend Team: Resolved.

L-02 | No Controls On Liquidation Fee Amount

Category	Severity	Location	Status
Validation	Low	Vanir.sol	Resolved

Description

During liquidation, the user pays an additional `feeAmount` from their collateral. This `feeAmount` is arbitrarily set by the admin within their `LiquidationRequest`, and can vary each time.

Without fee validation, the fee may be too small or too large, negatively impacting the protocol and user respectively.

Recommendation

Consider adding on-chain validations for `feeAmount` on liquidations, or clearly document the fee calculation behavior of your backend system.

Resolution

Norlend Team: We have a breakdown of the calculations documented in our backend.

L-03 | Failed Liquidations With Low Target LTV

Category	Severity	Location	Status
Warning	Low	Global	Resolved

Description

Norlend has a target LTV for liquidation within its backend system. If the LTV which Norlend is targetting has a large delta between the current LTV, all liquidation calls will fail as more amountIn is necessary than is approved and available for swap.

Recommendation

Clearly document this behavior and appropriately set the target LTV.

Resolution

Norlend Team: This is dependent on providing the correct parameters, we have found a target range that works for us and can adjust it to lesser deltas if needed.

L-04 | Failing Tests

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

The tests are meant to simulate Norlend's operations with current backend configurations, however numerous tests are failing. Some reasons for failure include but are not limited to:
1. Lack of pinned fork block number
2. Improper Target LTV's for partial liquidations
3. Incorrect `feeAmount` calculation within the tests

All tests should be passing prior to deployment.

Recommendation

Fix all tests and align them with Norlend's backend systems.

Resolution

Norlend Team: Acknowledged.

L-05 | No Events Emitted On Funds Rescue

Category	Severity	Location	Status
Event	Low	Vanir.sol	Acknowledged

Description

Two new functions has been added to the contract to enable the admin to reduce funds - ‘transfer’ and ‘transferEth’. There are no events emitted to notify for the action happening.

Recommendation

Consider if you should emit events in these two functions.

Resolution

Norlend Team: We have no needs for these events, we utilize a blockchain listener that tracks all Norlend-related activity.

L-06 | Frozen Pool Will Lead To Fail Liquidations

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

When the reserve configuration in SparkLend isFrozen, supplying liquidity is prevented by repays and withdraws are permitted. If the configuration were set to frozen, most Norlend liquidations would fail since function swap() attempts to supply liquidity that wasn't used as part of the swap cost.

Recommendation

Clearly document this risk.

Resolution

Norlend Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits