Flexlab

NORAH TOKEN ECOSYSTEM

Consolidated Smart Contract Security Audit Report

December 11, 2025

Flexlab

Table of Contents

1. Executive Summary 3
   1.1 Key Findings Summary 3
2. Audit Scope 4
   2.1 Contracts Audited 4
3. Critical Findings 5
   [C-01] Broken Bridging Logic in BridgeApplication 5
   [C-02] Transaction Mapping Collision in BridgeReserve 5
   [C-03] Double-Claiming of Rewards in QuarterlyRevenueObligation 5
   [C-04] Broken Consensus Logic in OracleIntegration 6
   [C-05] Broken Daily Limit Logic in NorahToken (Solana) 6
4. High Severity Findings 7
   [H-01] No Refunds on Cancellation or Failure (BridgeApplication) 7
   [H-02] Unsafe ERC20 Transfer Handling (BridgeReserve) 7
   [H-03] setChainReserve Resets State Dangerously (BridgeReserve) 7
   [H-04] Max Supply Circumvention (NorahToken EVM) 7
   [H-05] Missing Authority Initialization (Solana) 7
   [H-06] Bridge Reserve Allocation Not Enforced (Solana) 7
   [H-07] Missing Fund Transfers in FutureReceivablesAssignmentDeed 7
   [H-08] Centralized Oracle Data (CommodityBasedMinting) 8
   [H-09] Missing Cumulative Production Check (OutputRightsAgreement) 8
   [H-10] Single Point of Manipulation (OracleIntegration) 8
   [H-11] Decoupling of Token Ownership and Rights (TokenIssuanceAgreement) 8
   [H-12] Logical Error in createBeneficiaryRight (SPVFiduciaryRole) 8
5. Medium Severity Findings 9
   5.1 Centralization and Access Control 9
   5.2 Denial of Service Risks 9
   5.3 Logic and State Issues 9
6. Low Severity and Informational Findings 10
   6.1 Input Validation 10
   6.2 Code Quality 10
   6.3 Economic and Design 10
7. Prioritized Recommendations 11
   7.1 Immediate Actions (Pre-Deployment Blockers) 11
   7.2 High Priority (Before Mainnet) 11
   7.3 Medium Priority 11
   7.4 Low Priority 11
8. Conclusion 12

Flexlab

1. Executive Summary

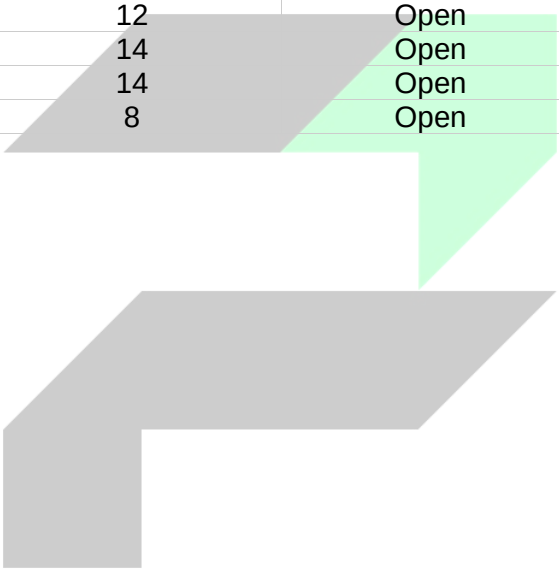
This consolidated report presents the findings of a comprehensive security audit performed on the Norah Token ecosystem smart contracts. The audit covered 15 contracts spanning the EVM (Solidity) and Solana (Anchor/Rust) blockchain platforms.

The Norah Token ecosystem is designed to manage a Real-World Asset (RWA) backed token with features including cross-chain bridging, quarterly revenue distributions, commodity-based minting, and comprehensive legal agreement tracking on-chain.

1.1 Key Findings Summary

The audit identified multiple critical vulnerabilities that pose immediate risks to user funds and system integrity. The most severe issues include broken bridging logic, double-claiming of rewards, transaction mapping collisions, and ineffective rate limiting mechanisms.

Severity	Count	Status	Impact
Critical	5	Open	Fund Loss / System Failure
High	12	Open	Major Disruption
Medium	14	Open	Conditional Risk
Low	14	Open	Minor Issues
Informational	8	Open	Best Practices



2. Audit Scope

The following 15 smart contracts were included in this security audit:

2.1 Contracts Audited

Contract	Platform	Risk Level
NorahToken.sol	EVM (Solidity)	Medium Risk
NorahToken (Solana)	Solana (Anchor)	Critical Risk
BridgeApplication.sol	EVM (Solidity)	Critical Risk
BridgeReserve.sol	EVM (Solidity)	Critical Risk
QuarterlyRevenueObligation.sol	EVM (Solidity)	Critical Risk
OracleIntegration.sol	EVM (Solidity)	Critical Risk
RobotMintManager.sol	EVM (Solidity)	Medium Risk
CommodityBasedMinting.sol	EVM (Solidity)	Medium Risk
TokenIssuanceAgreement.sol	EVM (Solidity)	Medium Risk
OutputRightsAgreement.sol	EVM (Solidity)	Medium Risk
NorahRWAIntegration.sol	EVM (Solidity)	Medium Risk
RevenueDistributionCovenant.sol	EVM (Solidity)	Medium Risk
SPVFiduciaryRole.sol	EVM (Solidity)	Medium Risk
FutureReceivablesAssignmentDeed.sol	EVM (Solidity)	Medium Risk
USDTMock.sol	EVM (Solidity)	Low Risk (Test Only)

3. Critical Findings

The following critical vulnerabilities require immediate attention before any production deployment. These issues pose significant risks to user funds and system integrity.

[C-01] Broken Bridging Logic in BridgeApplication

Contract: BridgeApplication.sol

Location: processBridge function

Description:

The processBridge function conflates incoming and outgoing bridge directionality. It validates incoming bridge transactions against the local bridgeRequests mapping which stores outgoing requests. This creates circular logic where bridging tokens effectively burns and re-mints them on the same chain.

Impact: Incoming bridging is impossible unless users have pending outgoing requests with matching nonces. Users pay fees to burn and immediately re-mint tokens, achieving no actual bridging.

Recommendation: Separate logic for incoming and outgoing bridges. Create a new executeIncomingBridge function that does not check bridgeRequests.

[C-02] Transaction Mapping Collision in BridgeReserve

Contract: BridgeReserve.sol

Description:

The contract uses a global bridgeTransactions mapping keyed by nonce, but nonces are generated per-user via userNonces[msg.sender]++. Multiple users will generate identical nonces (e.g., User A nonce 0, User B nonce 0), causing data overwrites.

Impact: Previous bridge transactions are overwritten. User funds can be erased from tracking, making finalization or refunds impossible.

Recommendation: Use keccak256(abi.encodePacked(msg.sender, nonce)) as the key, or implement a global nonce counter, or use nested mappings.

[C-03] Double-Claiming of Rewards in QuarterlyRevenueObligation

Contract: QuarterlyRevenueObligation.sol

Description:

The claimDistribution function determines user share based on current token balance at claim time rather than a historical snapshot. Users can claim rewards, transfer tokens to another account, and claim again.

Impact: Complete fund drainage. Malicious users can repeatedly claim until the contract's USDT balance is depleted, leaving honest late-claimers with nothing.

Recommendation: Implement historical balance snapshots using ERC20Votes.getPastVotes or ERC20Snapshot. Record snapshot block in calculateDistribution and use it in claimDistribution.

[C-04] Broken Consensus Logic in OracleIntegration

Contract: OracleIntegration.sol

Description:

The contract stores only one price entry per symbol globally. The calculateConsensusPrice function iterates over oracles but reads the same global value repeatedly, calculating an average of identical numbers.

Impact: Consensus mechanism is non-functional. System returns the last submitted price, enabling single-oracle manipulation.

Recommendation: Change storage to mapping(address => mapping(string => PriceData)) to store prices per oracle, then aggregate properly.

[C-05] Broken Daily Limit Logic in NorahToken (Solana)

Contract: norah_token (Solana/Anchor)

Description:

The daily limit check uses raw Unix timestamp (seconds) instead of day index. Since timestamps update every second, authority.last_mint_date != current_time is true for almost every transaction, resetting the daily limit per block instead of per day.

Impact: A compromised robot authority can mint up to daily_limit per block rather than per day, rapidly draining the supply.

Recommendation: Calculate day index: let current_day = Clock::get()?.unix_timestamp / 86400;

4. High Severity Findings

[H-01] No Refunds on Cancellation or Failure (BridgeApplication)

When requestBridge is called, tokens are burned. If the bridge is cancelled or failed, only the status updates - no tokens are refunded. Users permanently lose funds.

Recommendation: Implement refund mechanism via RobotMintManager or NorahToken to restore user balances.

[H-02] Unsafe ERC20 Transfer Handling (BridgeReserve)

SafeERC20 is imported but not properly applied. Direct transfer/transferFrom calls without return value checking could fail silently with non-standard tokens.

Recommendation: Use IERC20(address(norahToken)).safeTransferFrom() explicitly.

[H-03] setChainReserve Resets State Dangerously (BridgeReserve)

Calling setChainReserve for an existing chain resets availableReserve and pendingBridges to initial values, corrupting accounting of in-transit funds.

Recommendation: Separate initialization and update logic, preserving runtime values during updates.

[H-04] Max Supply Circumvention (NorahToken EVM)

receiveTokens (bridging) and _claimRewards (staking) call _mint directly without checking maxSupply, allowing supply to exceed intended cap.

Recommendation: Override _mint to enforce: require(totalSupply() + amount <= maxSupply)

[H-05] Missing Authority Initialization (Solana)

No instructions exist to initialize RobotAuthority or BridgeAuthority accounts. The contract is non-functional for minting operations.

Recommendation: Add administrative instructions to initialize and manage authority accounts.

[H-06] Bridge Reserve Allocation Not Enforced (Solana)

bridge_reserve_amount (30%) is calculated but never checked. robot_mint can consume 100% of supply, leaving nothing for bridging.

Recommendation: Enforce: current_supply + amount <= max_supply - bridge_reserve_amount for robot mints.

[H-07] Missing Fund Transfers in FutureReceivablesAssignmentDeed

executeDistribution records distributions without transferring any tokens. IERC20/SafeERC20 are imported but unused.

Recommendation: Implement actual token transfers or remove misleading imports and document off-chain settlement.

[H-08] Centralized Oracle Data (CommodityBasedMinting)

Chainlink oracle is initialized but never used. Owner manually sets price/yield values, enabling arbitrary minting manipulation.

Recommendation: Implement actual Chainlink integration or document centralized model clearly.

[H-09] Missing Cumulative Production Check (OutputRightsAgreement)

monetizeOutput/tokenizeOutput only check current transaction quantity against total, not cumulative. Operators can exceed rights indefinitely.

Recommendation: require(rights.totalOutputProduced + outputQuantity <= rights.outputQuantity)

[H-10] Single Point of Manipulation (OracleIntegration)

Due to shared storage, the last oracle to submit controls the entire system's price, bypassing redundancy safeguards.

Recommendation: Implement per-oracle storage and proper consensus aggregation.

[H-11] Decoupling of Token Ownership and Rights (TokenIssuanceAgreement)

Rights are not linked to actual token holdings. Users can sell tokens but keep rights, or issuers can create phantom rights.

Recommendation: Verify token balance during issuance and implement transfer hooks.

[H-12] Logical Error in createBeneficiaryRight (SPVFiduciaryRole)

beneficiary field is hardcoded to msg.sender (owner), making it impossible to assign rights to actual beneficiaries.

Recommendation: Add _beneficiary parameter to the function.

5. Medium Severity Findings

5.1 Centralization and Access Control

- [M-01] Centralized Validation - BridgeApplication accepts but ignores merkleProof, relying solely on authorizedValidators
- [M-02] Centralized Revenue Stream Creation - FutureReceivablesAssignmentDeed only allows owner to create streams

5.2 Denial of Service Risks

- [M-03] DoS in Data Updates - CommodityBasedMinting reverts data updates during mintingCooldown
- [M-04] Unbounded Loops - NorahRWAIntegration getSystemStatistics loops over all cycles
- [M-05] DoS via Array Growth - OracleIntegration removeOracle and getActiveOracles iterate unbounded arrays
- [M-06] Unbounded Loops - RevenueDistributionCovenant checkCovenantCompliance iterates three dynamic arrays
- [M-07] Unbounded Governance Calculation - TokenIssuanceAgreement updateGovernanceRights loops indefinitely

5.3 Logic and State Issues

- [M-08] Inefficient Token Burning - BridgeApplication uses two-step transfer then burn
- [M-09] Non-Functional Flash Loan Protection - NorahToken checks block.timestamp > 0 (always true)
- [M-10] Deprecated selfdestruct - NorahToken ToronetOwnable.destroy uses deprecated opcode
- [M-11] Invalid Interface Assumption - NorahRWAIntegration checkContractHealth assumes all contracts are ERC20
- [M-12] Race Condition - QuarterlyRevenueObligation processRevenue can be called after calculateDistribution
- [M-13] PDA Seed Collision - Solana NorahToken MintRequest limited to 1 per second
- [M-14] Strict Deadline Locks State - SPVFiduciaryRole prevents fulfillment after deadline with no resolution

6. Low Severity and Informational Findings

6.1 Input Validation

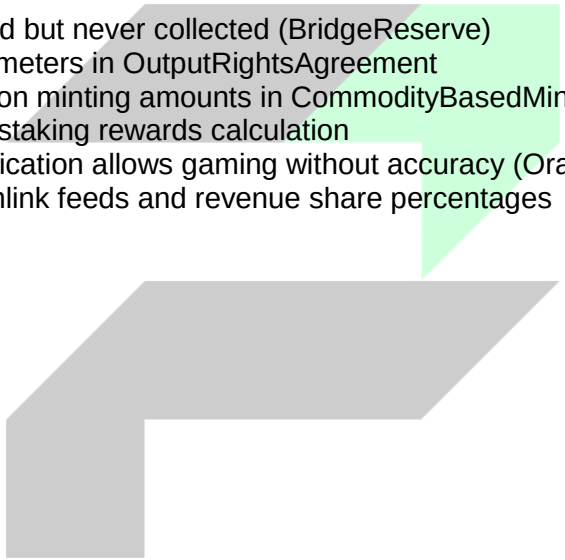
- Missing zero address checks in constructors (BridgeApplication, CommodityBasedMinting, SPVFiduciaryRole)
- Inconsistent input validation in NorahRWAIntegration updateSystemConfiguration
- Lack of validation for distribution amounts in FutureReceivablesAssignmentDeed

6.2 Code Quality

- Floating pragma (^0.8.17) - recommend locking version for production
- abi.encodePacked collision risk - recommend using abi.encode
- Unused imports (IERC20, SafeERC20) in multiple contracts
- Missing events for configuration updates
- Unused Chainlink Aggregator Interface in CommodityBasedMinting

6.3 Economic and Design

- Bridge fee defined but never collected (BridgeReserve)
- Unused fee parameters in OutputRightsAgreement
- No upper bound on minting amounts in CommodityBasedMinting
- Precision loss in staking rewards calculation
- Reputation gamification allows gaming without accuracy (OracleIntegration)
- Hardcoded Chainlink feeds and revenue share percentages



7. Prioritized Recommendations

7.1 Immediate Actions (Pre-Deployment Blockers)

- Fix BridgeApplication incoming/outgoing logic separation
- Fix BridgeReserve transaction mapping collision
- Implement historical snapshots in QuarterlyRevenueObligation
- Refactor OracleIntegration storage and consensus logic
- Fix Solana daily limit calculation (divide by 86400)
- Add authority initialization instructions to Solana program

7.2 High Priority (Before Mainnet)

- Implement refund mechanisms in BridgeApplication
- Apply SafeERC20 consistently across all contracts
- Enforce maxSupply globally in NorahToken _mint override
- Implement cumulative production tracking in OutputRightsAgreement
- Fix beneficiary assignment in SPVFiduciaryRole
- Link token ownership to rights in TokenIssuanceAgreement

7.3 Medium Priority

- Replace unbounded loops with counters or pagination
- Implement proper flash loan protection or remove misleading code
- Remove deprecated selfdestruct functionality
- Implement actual Chainlink integration or document centralization
- Add merkle proof verification or remove unused parameter

7.4 Low Priority

- Lock pragma versions for production
- Add comprehensive zero-address validation
- Remove unused imports and code
- Add events for all configuration changes
- Use multi-sig wallets for all admin functions

8. Conclusion

[PUBLIC RELEASE VERSION]