Asfalia

Security Audit Report

Shield Finance

Asfalia Audit on November 27, 2025

Table of Contents

Summary

This report has been prepared for Shield Finance to discover issues and vulnerabilities in the source code of the Shield Finance project as well as any contract dependencies that were not part of an officially recognized library.

A comprehensive examination has been performed, utilising Static Analysis and Manual Review techniques.

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.

The security assessment resulted in findings that ranged from Medium to informational.
We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Shield Finance
Platform	EVM
Chain	Ethereum Smart Chain
Language	Solidity
Codebase	Files provided (closed source)
Commit	N/A

Audit Summary

Delivery Date	11/27/2025
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Pass

Project Overview

The ShieldToken project introduces SHIELD, a fixed-supply ERC20 token designed for a simple and transparent fair-launch model. The contract is intentionally minimal, relying exclusively on well-audited OpenZeppelin libraries to ensure predictable behavior and reduce the risk of vulnerabilities. SHIELD implements no taxes, no blacklist mechanisms, no trading controls, and no privileged minting functions, making it a pure, immutable ERC20 token after deployment.

A total supply of 10,000,000 SHIELD is minted at deployment and assigned entirely to the deployer, who is responsible for all subsequent distribution, liquidity provision, or community allocation. The token includes optional burn functionality, allowing holders to reduce circulating supply voluntarily, supporting community-driven deflation if desired.

StakingBoost Contract

Status

Pass

Scope:

This audit covers the smart contract **StakingBoost,** written in Solidity ^0.8.20. The scope includes staking logic, reward distribution mechanics, FXRP reward handling, boost calculation, administrative controls, token recovery, and interactions with external contracts ***(IERC20, IShXRPVault)***.

The review focuses solely on on-chain behavior, trust assumptions, and systemic risks arising from contract design, not external infrastructure (e.g., vault implementation, revenue pipeline, token behavior).

Objective:

The objective of this audit is to identify vulnerabilities, logic inconsistencies, centralization risks, reentrancy exposure, trust assumptions, and token-handling issues that may affect correctness, economic safety, or user funds.

The audit evaluates the contract's resilience against common attack vectors, its compliance with intended staking/reward semantics, and the potential impact of owner privileges or misconfiguration.

Findings:

Below are the enumerated issues identified during the security review, each categorized by severity.

1. Centralized FXRP Reward Control (High Severity)

The contract allows the owner to withdraw all FXRP via recoverTokens, including FXRP implicitly owed to users as pending rewards. This introduces a trust-based model where the owner can remove the full reward pool, preventing future claims and enabling a reward-level rug pull. User principal (SHIELD) remains protected, but rewards do not.

2. Missing Reentrancy Protection on recoverTokens (Low Severity)

recoverTokens is not nonReentrant. If the owner recovers a malicious ERC-20, that token could reenter the contract and trigger other functions unexpectedly. While this requires an owner-initiated action and a malicious token, it remains a deviation from best practice and should be patched.

3. Lock Period Applies Per Account, Not Per Deposit (Informational)

stakedAt is only set on the first stake and not refreshed with subsequent stakes. Users can deposit additional SHIELD and withdraw all of it once the first 30-day period expires. This behavior may not align with user expectations of "every deposit is locked for 30 days."

4. Assumption of Standard ERC-20 Tokens (Medium Severity)

The system does not support fee-on-transfer or rebasing tokens. If SHIELD or FXRP behaves non-standardly, accounting mismatches can occur (e.g., totalStaked > actual balance). This could break withdrawals or reward distribution.

5. Orphaned FXRP Rewards When No Stakers Exist (Low Severity)

If distributeBoost is called while totalStaked == 0, FXRP remains trapped in the contract and can only be reclaimed by the owner. This may create perceptions of reward unfairness or misallocation.

6. Non-Robust Approval Pattern for FXRP (Low Severity)

The contract uses raw approve() instead of SafeERC20 helpers or the safe zero-reset pattern. Non-standard tokens (e.g., USDT-style) may revert or fail silently, leading to failed reward claims.

7. Missing Zero-Address Validation for Revenue Router (Low Sever-

The constructor does not validate that _revenueRouter is non-zero, risking deployments where reward distribution becomes impossible.

StakingBoost Contract

Status
Pass

Conclusion: The **StakingBoost** contract is generally well-structured, uses safe patterns for staking and Synthetix-style reward accounting, and protects user principal effectively.
The primary concern is ***centralization*** of ***FXRP reward control***, which places substantial trust in the owner. Secondary issues relate to token assumptions, approval robustness, and UX expectations around lock periods.

Recommendations:
- Restrict FXRP Recovery or Only Allow Recovery of “Excess” FXRP to prevent draining reward pools.
- Add nonReentrant to recoverTokens to ensure uniform reentrancy safety.
- Clarify lock semantics or update logic to reflect intended behavior (per-deposit vs per-account).
- Use SafeERC20.safeIncreaseAllowance or zero-reset approve pattern for compatibility.
- Add constructor validation for _revenueRouter != address(0).
- Document token assumptions clearly (non-rebasing, non-fee-on-transfer).

ShieldToken

Status
Pass

Objective: The objective of this audit is to evaluate the ShieldToken contract for vulnerabilities, trust risks, logical inconsistencies, and potential attack surfaces. The review focuses on ensuring that token minting, burning, ownership, and ERC20 functionality behave as intended with no pathways for unauthorized minting, supply manipulation, or privilege escalation. The goal is to provide confidence that the contract is safe, predictable, and adheres to standard ERC20 practices.

Findings:

1.Centralized Token Distribution & “Fair Launch” Claim – Informational

All tokens are minted to the deployer address at launch. While not a security vulnerability, this design creates trust implications around distribution fairness. Token fairness is dependent entirely on the deployer’s actions, which may impact community perception.

2. Unused Ownable Inheritance – Informational

The contract inherits from Ownable, yet no owner-only functions exist. While this does not create technical risk, the presence of an owner may raise transparency questions from users who expect an admin-free token.

3. TOTAL_SUPPLY Constant vs totalSupply() – Informational

The contract defines a fixed TOTAL_SUPPLY constant but still relies on the ERC20 internal supply variable. This is safe as long as no future minting capability is added; however, the constant could become misleading in extended versions.

4. OpenZeppelin Version Dependency – Informational

The constructor uses the OpenZeppelin v5 pattern (Ownable(msg.sender)). If the library version is changed, the contract may fail to compile or behave inconsistently. Explicit version pinning is recommended to prevent future integration issues.

5. Minor Style & Gas Micro-Optimizations – Low

Minor stylistic improvements (such as using 1e18 instead of 10**18) could improve readability but do not affect security or functionality.

ShieldToken

Status
Pass

Conclusion:

Total Issues Identified:

Critical: ***0***
High: ***0***
Medium: ***0***
Low: ***1***
Informational: ***4***

The ShieldToken contract is structurally sound, minimal, and leverages well-audited OpenZeppelin modules. No vulnerabilities were found that could compromise token balances, supply integrity, or transfer mechanics. All identified findings are non-critical and relate to ecosystem trust, maintainability, or code clarity.

Recommendation:

- Remove or renounce ownership if the project aims for a fully trustless launch.
- Clearly communicate token distribution plans to address fairness expectations.
- Pin the OpenZeppelin version used for deployment.
- Optionally refactor minor stylistic elements for clarity.

Overall, the contract is safe, predictable, and suitable for deployment as a standard fixed-supply ERC20 token.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Asfalia’s prior written consent in each instance.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 Asfalia to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-freenature 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. Asfalia’s position is that each company and individual are responsible for their own due diligence and continuous security. Asfalia’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 Asfalia 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.

Project is potentially vulnerable to 3rd party failures of service - namely in the form of APIs providing the price for the currencies used by the project. Project could become at risk if these APIs provided incorrect pricing.

Audit does not claim to address any off-chain functions utilized by the project.

The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone.

With over 30 years of combined experience in the DeFi space, our team is highly dedicated to delivering a product that is as streamlined and secure as possible. Our mission is to set a new standard for security in the auditing sector, while increasing accessibility to top tier audits for all projects in the crypto space. Our dedication and passion to continuously improve the DeFi space is second to none.