IMMUNEBYTES

Security Assessment

Smart Contract Audit

Collect Foundation

29th Nov 2025

collectfoundation.xyz
v1.0

Introduction

Name	Collect Foundation
Website	https://collectfoundation.xyz/
Repository/Source	0xeDbba828A5Aa774a1617db584cBa79c4AC85d710
Platform	L1
Network	Ethereum and EVM compatible chains
Language	Solidity
Commit Hash	NA
Timeline	27th Nov 2025 - 28th Nov 2025

Table of Content

About Collect Foundation

CollectFoundation.xyz is a Web3-oriented platform that markets itself as a place to vault, trade, authenticate, and manage physical and digital collectibles (like Pokémon cards, comic books, and other trading cards) with blockchain-based ownership and trading features. It positions itself as bridging physical collectibles with on-chain digital tracking and trading, and links to apps and marketplaces (e.g., Fanable) where users can buy, sell, and explore collectibles.

Executive Summary

ImmuneBytes conducted a focused security review of TokenFixed, a minimal fixed-supply ERC-20 intended for a lightweight token launch and basic UI integration. The contract's logic is simple and does not include complex value-moving flows, upgrade mechanisms, or privileged minting beyond initial deployment. No critical or high-risk vulnerabilities were identified.

A few notable user-facing concerns are due to missing documentation and reliance on implicit assumptions. The published maxSupply does not match the unit scale of totalSupply, which can mislead users, explorers, and exchanges about actual supply and valuation. Additionally, the owner retains authority to withdraw any ERC-20 sent to the contract—appropriate for recovery, but easily misunderstood without clear policy.

Overall, TokenFixed is technically sound for a simple token deployment, but transparency around supply semantics, vault behavior, and owner authority is essential to avoid user misconception, reputational risk, and trust-model ambiguity in production environments.

System Architecture

TokenFixed is a simple ERC-20 that mints its entire supply to a designated vault contract upon deployment. The token inherits ERC20Burnable, enabling holders to reduce supply, and grants the owner the exclusive ability to withdraw arbitrary ERC-20 tokens sent to the contract.

The constructor enforces that the vault is a contract using AddressUtils.isContract and mints _maxSupply * 1e18 units, storing maxSupply in unscaled form. The token itself has no mint function beyond initialization, making supply effectively fixed post-deployment.

The owner may call adminTokenWithdraw to sweep any ERC-20 tokens held by the TokenFixed contract. Apart from this, the system has no upgradeability, no fee logic, no governance hooks, and no further value-moving operations.

Trust Assumptions

- Vault contract is trusted to hold and manage the full initial supply.
- The Owner is trusted to act responsibly when using adminTokenWithdraw.
- Users are expected not to send tokens to the contract address except intentionally.
- AddressUtils.isContract correctly identifies deployed contracts (not those under construction or self-destructed).
- No external documentation exists to clarify supply semantics or vault behavior.

System Security Concerns

Supply clarity and public metadata
Public supply parameters should match unit conventions of totalSupply to prevent misinterpretation by explorers, users, and exchanges. Misalignment impacts valuation and market transparency.

Vault centralization and custodial opacity
Users depend on the vault's correctness, lifecycle, and permissions. A contract-only requirement without visibility into the vault's code introduces custodial opacity.

Owner token-rescue authority
A broad sweeping function allows the owner to move any ERC-20 from this contract, including this token. While intended for safety, without documentation it can be misinterpreted as arbitrary control.

Token accident-handling and user expectations
Users frequently send tokens to incorrect addresses. Without strict documentation, the behavior of refunding or sweeping becomes a potential reputational and user-trust liability.

AddressUtils behavioral limits
The extcodesize-based checks misclassify contracts under construction or destroyed contracts, affecting vault expectation accuracy.

Security Checklist

During our comprehensive audit of the Ethernity project, we have assessed the following potential vulnerabilities inspired by common issues encountered in Ethereum-based systems:

Access Control Issue	✅
Initialization Vulnerabilities	✅
Reentrancy Attacks	✅
Logic and Calculation Errors	✅
Unintended State Manipulation	✅
ERC20 Compliance	✅
Token Swapping Risks	✅
Denial of Service (DoS)	✅
Ownership Validation	✅
Integer Overflow and Underflow	✅
Error Handling	✅
Frontrunning Risks	✅
Audit-Specific Issues	✅
Merkle Tree and Leaf Nodes	✅

Methodology

We performed a manual, invariant-driven analysis of TokenFixed, reviewing constructor flows, value distribution, supply initialization, authority boundaries, and ERC-20 interaction surfaces. Our assessment includes line-by-line evaluation of mint logic, owner authority, external token transfers, vault dependency, and the AddressUtils library, extraction of implied invariants from code behavior in the absence of external specification, review of publicly exposed fields for potential user-misinterpretation (maxSupply, totalSupply), analysis of trust assumptions introduced by the vault contract and the owner's rescue function, threat modeling around accidental user behaviors, token transfer errors, ownership centralization, and economic misalignment. No formal verification or fuzzing was performed. Recommendations emphasize transparency, supply-clarity, and user-safety in low-documentation deployment contexts.

Security Review Report

Severity	Open	Acknowledged	Partially Resolved	Resolved	TOTAL
Critical	-	-	-	-	-
High	-	-	-	-	-
Medium	-	-	-	-	-
Low	-	4	-	-	-
Informational	-	-	-	-	-
TOTAL	-	4	-	-	-

Severity Reference

1. Critical
   Issues causing protocol insolvency, unauthorized theft, or irreversible loss or alteration of assets, or governance outcomes.

2. High
   Issues involving theft or freezing of unclaimed yield, royalties, or temporary locking of assets.

3. Medium
   Operational failures, inefficiencies, or exploitations causing delays, excessive gas usage, or disruption without direct loss.

4. Low
   Minor deviations in promised functionality without impact on underlying value.

5. Informational
   Recommendations for code readability, maintainability, or best practices.

Status Reference

1. Open
   The issue has been identified and is pending resolution.

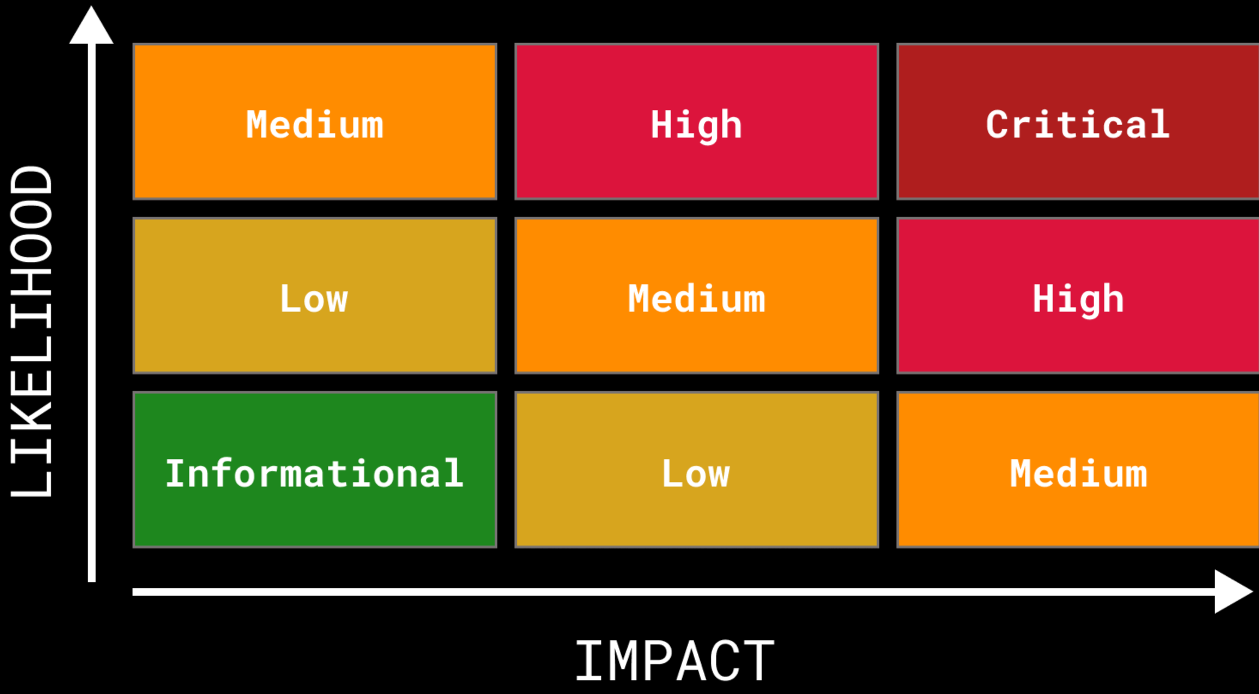
2. Acknowledged
   The issue has been reviewed and accepted by the team but remains unresolved.

3. Partially Resolved
   The issue has been mitigated to some extent, but residual risks or concerns persist.

4. Resolved
   The issue has been fully addressed, with no further action required.

5. Closed
   The issue is no longer relevant or actionable, regardless of resolution.

Risk Matrix


Medium
High
Critical
Low
Medium
High
Informational
Low
Medium
LIKELIHOOD
IMPACT

Summary of Findings

CODE	ISSUE	SEVERITY	STATUS
CF01	Owner Can Withdraw Any Token Held by the Contract	Low	Acknowledged
CF02	Misrepresented maxSupply Relative to totalSupply Units	Low	Acknowledged
CF03	No On-Chain Enforcement of “Fixed Supply” Guarantee	Low	Acknowledged
CF04	Behavioral Limits of AddressUtils May Mislead Vault Expectations	Low	Acknowledged

Findings Overview

Severity: Low	
CF01	**Owner Can Withdraw Any Token Held by the Contract**
Status	Acknowledged

Description

The owner can unilaterally withdraw any ERC-20 token from the TokenFixed contract, including the native token. While intended for recovering mistakenly sent funds, the absence of documentation makes this indistinguishable from arbitrary asset extraction. Users interacting with the token contract address (accidentally or via poor UI design) may lose funds without understanding why.

Impact

- Users: accidental transfers can be seized, leading to monetary loss.
- Protocol: centralization optics; potential accusations of administrative overreach.
- Exchanges/trackers: classify the token as high-trust-required or admin-custodial.

Code Block Affected

```
function adminTokenWithdraw(address _tokenAddress, uint256 _amount) public onlyOwner {

        IERC20 token = IERC20(_tokenAddress);

        SafeERC20.safeTransfer(token, owner(), _amount);

}
```

Recommendation

- Emit detailed withdrawal events.
- Optionally restrict withdrawals of this token to avoid misinterpretation.
- Publicly document rescue semantics for user awareness.

Severity: Low	
CF02	**Misrepresented maxSupply Relative to totalSupply Units**
Status	Acknowledged

Description

The maxSupply variable is stored unscaled while totalSupply reflects _maxSupply * 1e18. This inconsistency propagates incorrect supply values to explorers, pricing tools, and user dashboards. In the absence of documentation, users may misinterpret supply, market cap, or inflation guarantees.

Impact

Users: misleading supply perception, poor investment decisions.

Protocol: distrust during listings, perceived tokenomics inconsistency.

Indexers: inaccurate market-cap, supply, and valuation metrics.

Code Block Affected

```
_mint(_vault, _maxSupply * (10 ** 18));

maxSupply = _maxSupply;
```

Recommendation

- Store maxSupply in base units (scaled).
- Or rename to maxSupplyUnits and publicly clarify the meaning.
- Align supply-related events and metadata with common ERC-20 conventions.

Severity: Low	
CF03	**No On-Chain Enforcement of “Fixed Supply” Guarantee**
Status	Acknowledged

Description

The contract offers no mint functionality post-deployment, but the stored maxSupply variable is not bound to any enforceable invariant. Without documentation, users may incorrectly assume the existence of cryptographic supply caps or governance-backed constraints.

Impact

- Users: misunderstanding of supply guarantees.
- Protocol: unnecessary skepticism during token review.
- Integrators: conflicting supply metadata.

Recommendation

- Document supply-cap behavior.
- Or remove maxSupply entirely to eliminate ambiguity.

Severity: Low	
CF04	**Behavioral Limits of AddressUtils May Mislead Vault Expectations**
Status	Acknowledged

Description

The extcodesize returns zero for contracts in construction or destroyed contracts. A vault under construction may be misclassified as an EOA; a vault destroyed later loses contract identity. Without documentation, users may assume stronger guarantees than actually provided.

Impact

- Users: false assumptions about vault immutability.
- Protocol: brittle deployment if using factory-pattern vaults.
- Integrators: inconsistent vault classification.

Recommendation

- Clarify the limitations of isContract in the documentation.
- Optionally enforce vault immutability via known-good deployment patterns.

.

Disclaimer

ImmuneBytes' security audit services are designed to help identify and mitigate potential security risks in smart contracts and related systems. However, it is essential to recognize that no security audit can guarantee absolute protection against all possible security threats. The audit findings and recommendations are based on the information provided during the assessment.

Furthermore, the security landscape is dynamic and ever-evolving, with new vulnerabilities and threats emerging regularly. As a result, it is strongly advised to conduct multiple audits and establish robust bug bounty programs as part of a comprehensive and continuous security strategy. A single audit alone cannot protect against all potential risks, making ongoing vigilance and proactive risk management essential.

To ensure maximum security, the project must implement the recommendations provided in the audit report and regularly monitor its smart contracts for vulnerabilities. It is imperative to adopt appropriate risk management strategies and remain proactive in addressing potential threats as they arise.

Please note that auditors cannot be held liable for any security breaches, losses, or damages that may occur after the audit has been completed despite using audit services. The responsibility for implementing the recommendations and maintaining the ongoing security of the systems rests solely with the project.