IMMUNEBYTES

Security Assessment

Smart Contract Audit

Epic

28th Mar 2025

Epic Chain
epictoken
v2.0

Introduction

Name	Epic Chain
Website	https://www.epicchain.io/
Repository/Source	https://github.com/epiconchain/epictoken/tree/swapupdate
Platform	L1
Network	Ethereum and EVM compatible chains
Language	Solidity
Commit Hash	7cb512d1a4ab085c72ef5c906690d41add1c2f75
Timeline	25th Mar 2025 - 28th Mar 2025

Table of Content

Executive Summary

This feature audit report covers the security review of a feature implementation for the Epic Token Swap contract. The audit commenced between 25th and 28th March 2025, whereas, this report captures the observations from the final day of the ImmuneBytes security team's assessment of the scoped contract.

The v2 update of the TokenSwap contract introduces several critical enhancements designed to address vulnerabilities observed in previous versions and mitigate risks exposed by high-profile incidents, such as the Wintermute hack. Key improvements include the integration of pausable functionality, the implementation of a Merkle Tree-based whitelist to restrict swap participation, and a robust check on outgoing token balances prior to token transfers. These modifications aim to bolster security, enhance operational control, and ensure that token swap operations are executed reliably and transparently.

This audit focuses on the newly added functionalities and modifications in the v2 TokenSwap contract. Specifically, the scope covers the verification of the Merkle Tree-based whitelist,

- ensuring that only authorized wallets can execute swaps;
- the evaluation of pausable functionality to guarantee rapid emergency response and controlled contract state transitions;
- and the validation of outgoing token balance checks that prevent the execution of swaps exceeding available supplies.

Additionally, the audit assesses that no regressions or unintended vulnerabilities were introduced with these new features.

Project Overview

Key Components

1. Initialization
   - Secure initialization ensures that only authorized entities can set up critical contract parameters such as the governor, mintable contract, and token swap configurations.
   - Proper access control checks are implemented to prevent unauthorized initialization of sensitive contract states.

2. Voting and Governance
   - Ensure the voting mechanism is robust and accurately tracks the votes required for executing key actions like minting.
   - Prevent scenarios where vote manipulation could disrupt the governance logic.
   - Validate that vote resetting after successful operations functions as intended.

3. Minting Functions
   - Enforce the 12-month minting wait period and the 12% supply cap for new tokens.
   - Validate governor restrictions, ensuring only the designated governor contract can mint new tokens.
   - Verify that unauthorized entities cannot modify mintable contract parameters or exploit minting logic.

4. Token Swap
   - Ensure the swap process securely handles incoming and outgoing tokens and adheres to burn mechanics.
   - Validate the burn address and token swap configurations for correctness and security.
   - Prevent misuse of the token swap contract by ensuring all configuration changes are properly authorized.

5. Admin Functions
   - Verify that only authorized entities, such as the multisig owner, can perform sensitive admin actions like setting the governor, token swap addresses, or outgoing tokens.
   - Confirm the integrity of admin-configurable parameters to prevent unintended state changes or inconsistencies.
   - Validate the correctness of modifiers like onlyOwner and onlyGovernor.

6. Reentrancy and Security
   - Ensure all state-changing functions are protected against reentrancy attacks.

7. Secure Merkle Tree Usage
   - Verify that the contract implements robust checks to store, validate and utilize the merkle roots appropriately.

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

ImmuneBytes team has performed thorough testing of the project, starting with analyzing the code design patterns where the smart contracts and provided technical documentation were reviewed to ensure the architecture of the codebase aligns with the business specifications from a bird's eye view. During this phase, the invariants were identified along with execution of the pre-existing test suite to ensure smooth working of the implemented functionality and set up for the auditor's edge case testing in upcoming phases

Our team then performed a formal line-by-line inspection of the Smart Contract to find potential issues like Signature Replay Attacks, Unchecked External Calls, Merkle Tree Exploits, Reentrancy, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the next phase, the team ran curated test cases in order to verify the findings and issues identified during previous phases so as to provide a proof of concept. The team also runs automated tests over the provided code base using a combination of external and in-house-developed tools to identify vulnerabilities and security flaws of static nature.

The code was audited by a team of independent auditors, which included -

- Testing the functionality of the Smart Contracts to determine proper logic has been followed throughout.
- Thorough, line-by-line review of the code, done by a team of security researchers with experience across Blockchain security, cybersecurity, software engineering, game theory, economics and finance.
- Deploying the code on localnet using open source testing frameworks and clients to run live tests.
- Analyzing failing transactions to check whether the Smart Contracts implementation handles bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest secure version.
- Analyzing the security of the on-chain data and components.

Scope

In-Scope Files

The following files and their respective locations were included in the scope of the audit:

1. *epictoken/tree/governor/contracts/TokenSwap.sol*
    - Token swapping contract uses Merkle Tree Roots to whitelist allowed wallets. These wallets are facilitated to perform exchanges between tokens, with mechanisms for burning the incoming token and ensuring liquidity for outgoing tokens.

Critical Area of Focus

1. Merkle Tree Whitelist Implementation: Confirm that the whitelist mechanism correctly verifies eligible wallets using Merkle proofs, thereby restricting unauthorized access to swap functionality.

2. Pausable Functionality: Ensure that the pause and unpause mechanisms work as intended, allowing for swift operational halts in emergency situations without compromising contract integrity.

3. Outgoing Token Supply Verification: Validate that the contract checks its balance for the outgoing token before processing swaps, preventing under- or over-allocation during token transfers.

4. Overall System Resilience: Examine the integration of these new features to confirm that they do not introduce regressions or new vulnerabilities, maintaining robust security and consistent performance under various operational scenarios.

Out of Scope

The following areas were excluded from the scope of this audit:

- Frontend Integrations: The review did not include the user interface or frontend applications that interact with the Ethernity contracts.

- External Contracts: Integration with external smart contracts or decentralized applications (dApps) beyond the provided codebase.

- Deployment Process: Gas optimization strategies and deployment processes were not examined.

Security Review Report

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

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.

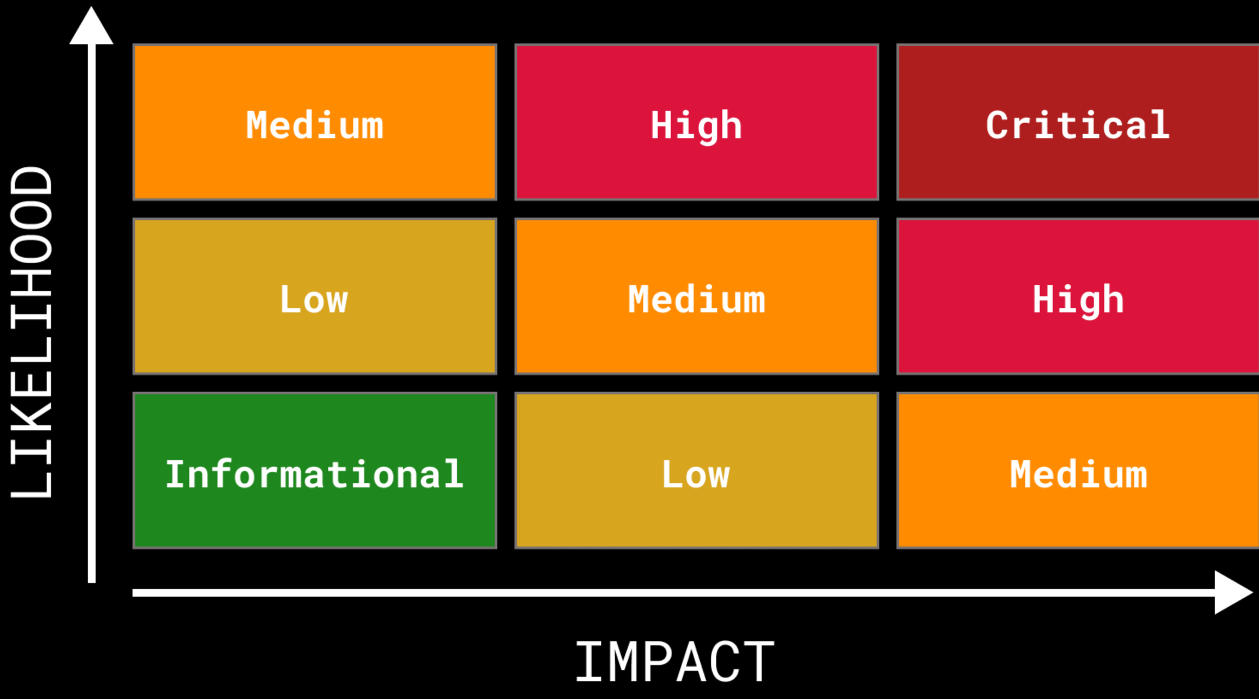
2. Acknowledged
   The issue has been reviewed and accepted by the team but remains unresolved.

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

No Issues Found

Our comparative analysis of the v2 release shows that the newly introduced features have effectively addressed recently observed novel security concerns, pertaining to which no new vulnerabilities have been identified. The updated contract exhibits a strong security posture with improved validation mechanisms and enhanced control flows. Overall, the enhancements contribute to a more resilient system, aligning with best practices while mitigating risks associated with unauthorized swaps and misconfiguration.

Disclaimer

STAY AHEAD OF THE SECURITY CURVE.