Ethernity Chain

Smart Contract Audit Report

IMMUNE BYTES

Audits

March 8, 2021

Introduction

1. About Ethernity Chain

Ethernity is a Decentralized Application (DAPP) Platform that allows artists to create and auction artwork inspired and backed by celebrities for charity.
The concept behind Ethernity is mutually beneficial for all actors involved:

1. **Public Figure:** by making it easier to create, store, back, and sell the artworks.
2. **Charity:** by getting 100% of the first sale proceeds (minus exchange fees). And the auction format maximizes the artwork value (increasing the charity's benefits) without the need of a promoter, leveraging the emotions that a bidding war involves.
3. **Collector:** by providing them with an easy, democratized platform to bid on these pieces of authentic digital art where they can thereafter take bids and auction their acquired artwork.

Visit https://ethernity.io/ to know more.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 15+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The Ethernity team has provided documentation for the purpose of conducting the audit. The documents are:

1. Ethernity White Paper v0.3

Audit Process & Methodology

ImmueBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

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

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Ethernity
- Languages: Solidity(Smart contract)
- Code for audit (Mainnet): 0xBBc2AE13b23d715c30720F079fcd9B4a74093505
- Code for audit (Kovan): 0x3634528B0D2BA04E7e678e9b6812EFd414aFC8c8

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level References

Every issue in this report was assigned a severity level from the following:

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	2
Closed	-	-	2

Low severity issues

1. **Redundant _decimals variable.**
   In **ERNToken** smart contract, a **uint _decimals** variable is used to store the token decimals value. However, a similar **uint8 _decimals** variable is also used in OpenZeppelin's **ERC20** contract. The value of both variables is 18. It is never recommended to use multiple variables to perform the same task.

   Recommendation:
   Consider removing the **uint _decimals** variable from the **ERNToken** contract.

2. **Use _*msgSender*() instead of *msg.sender***
   Throughout the **ERNToken** contract **_msgSender**() is used to access the caller's address instead of **msg.sender** except for one place, in the **constructor**() **Line 608 msg.sender** is used. To maintain consistency it is always recommended to use exact same conventions throughout the contract.

   Recommendation:
   Consider replacing **msg.sender** with **_msgSender**() inside the **constructor**().

3. **Different Initial Supply was minted than the one mentioned in the whitepaper.**
   The **Whitepaper** of Ethernity, **The $ERN Token section (page 9)** mentions that an initial supply of 55 Million $ERN Tokens will be minted while as can be seen on etherscan an initial supply of 30 Million $ERN Tokens were minted.

   Acknowledged(March 8th, 2021): The issue is acknowledged by the Ethernity team. The specification has been changed in the final version of the whitepaper.

4. **Missing stake and rewards functions as mentioned in the whitepaper.**
   In the **Whitepaper** of Ethernity, **The Technical Design - Overview** section, **bullet point #3 (page 10)** mentions that "*The $ERN token is an ERC-20 token with stake functions*" and the **Smart Contracts** section **bullet point #1** mentions that *"The $ERN token will be an OpenZeppelin ERC20 based token with stake and rewards functions*". However, no such functions are present in the **ERNToken** smart contract.

   Acknowledged(March 8th, 2021): The issue is acknowledged by the Ethernity team. The specification has been changed in the final version of the whitepaper.

Unit Test

No unit tests were provided by the Ethernity team.

Recommendation:
Our team suggests that the developers should write more extensive test cases for the contracts.

Coverage Report

Coverage report cannot be generated without unit test cases.

Recommendation:
We recommend 100% line and branch coverage for unit test cases.

Concluding Remarks

While conducting the audits of the Ethernity smart contract, it was observed that the contracts contain only Low severity issues.

Resolving the areas is up to Ethernity's discretion. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Ethernity platform or its product neither this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the one audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes Pvt Ltd.