Ethernity Chain

Smart Contract Audit Report

June 2023

Disclaimer

This disclaimer is to inform you that the report you are reading has been prepared by Mantisec Labs for informational purposes only and should not be considered as investment advice. It is important to conduct your own independent investigation before making any decisions based on the information contained in the report. The report is provided "as is" without any warranties or conditions of any kind and Mantisec Labs excludes all representations, warranties, conditions, and other terms. Additionally, Mantisec Labs assumes no liability or responsibility for any kind of loss or damage that may result from the use of this report.

It is important to note that the analysis in the report is limited to the security of the smart contracts only and no applications or operations were reviewed. The report contains proprietary information, and Mantisec Labs holds the copyright to the text, images, photographs, and other content. If you choose to share or use any part of the report, you must provide a direct link to the original document and credit Mantisec Labs as the author.

By reading this report, you are accepting the terms of this disclaimer. If you do not agree with these terms, it is advisable to discontinue reading the report and delete any copies in your possession.

Audit Goals

Mantisec Labs was commissioned by the Ethernity team to perform an audit of their smart contracts. The audit was conducted in the month of **June 2023.**

The purpose of this audit was to achieve the following:

i. Identify potential security issues within the smart contract

ii. Formally check the logic behind the given smart contract

Information in this report should be used for understanding the risk exposure of this smart contract, and as a guide to improving the security posture by remediating the issues that were identified.

Audit Details

- Project Name: Ethernity Chain
- Contract Name: TeamVesting.sol
- Languages: Solidity(Smart contract)
- Platforms and Tools: Remix IDE, Solhint, VScode, Contract Library, Slither

Security Level Reference

Every issue in this report were 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	0	0	0

Closed	0	1	1

Contract Name: TeamVesting.sol

Medium Severity issues

Zero Division Error Prevention in addAccount Function

Description:

The function addAccount does not include a **require** statement to ensure that **_vestingDays** is a value greater than zero. This may result in a zero division error in the allowance function if **_vestingDays** is zero when computing the daily allowance.

Recommendation:

By including the require statement **require(_vestingDays > 0, "Vesting days must be greater than zero");**, the function ensures that **_vestingDays** is a positive value and prevents any zero division errors that might occur in the **allowance** function when calculating the daily allowance

Low Severity Issues

Reentrancy Guard Implementation in withdraw Function

Description:

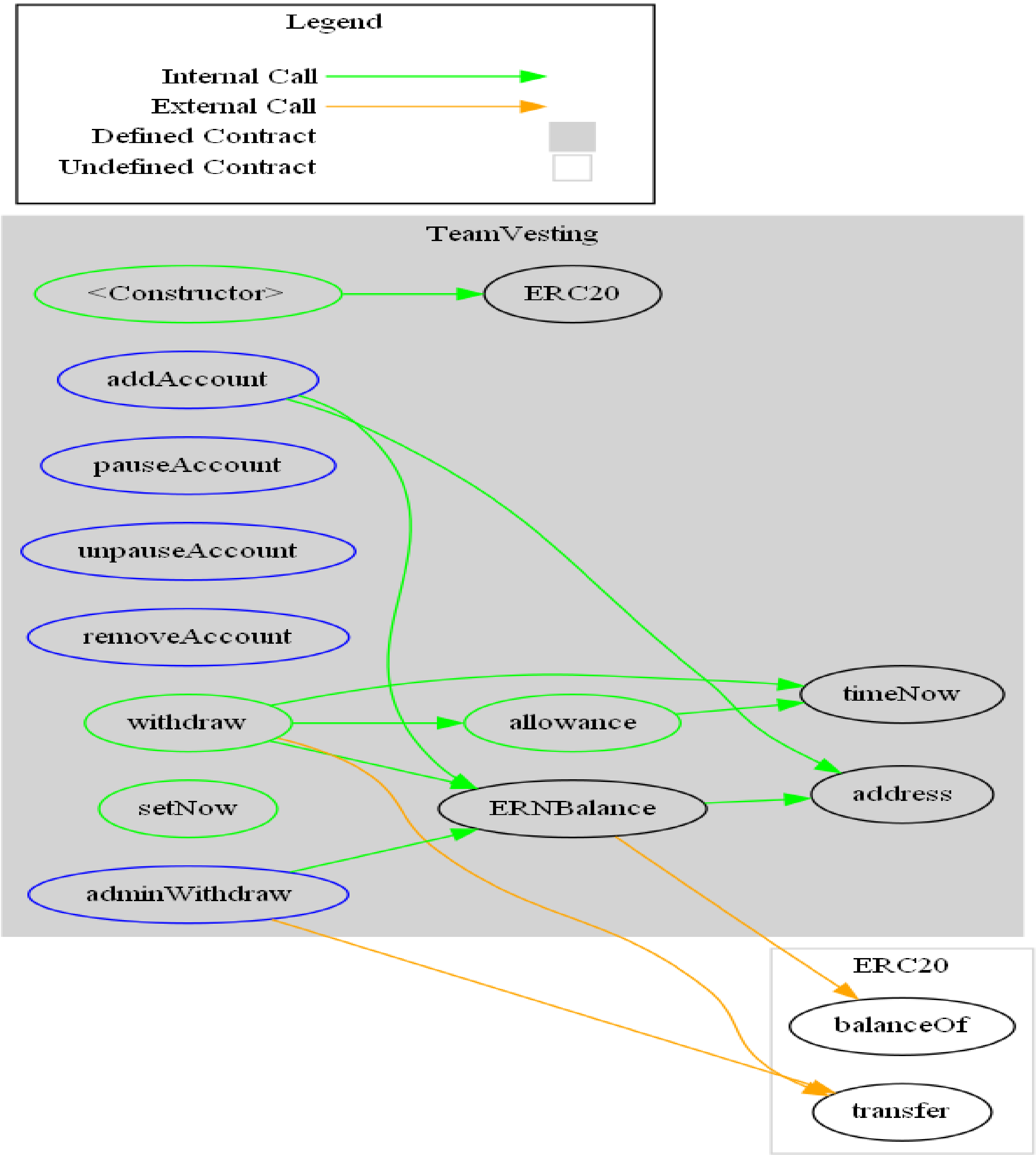
The code does not include a reentrancy guard for the **withdraw** function, which may leave the contract vulnerable to reentrancy attacks. While the state is updated prior to the token transfer, it is advisable to implement a reentrancy guard as a precautionary measure to avoid reentrant calls during the function's execution.

Recommendation:

To mitigate the risk of reentrancy attacks, it is recommended to include a reentrancy guard in the **withdraw** function.

Additional Details

1. **TeamVesting.sol**



Concluding Remarks

While conducting the audit of the TeamVesting smart contract, it was observed that the contracts contained High,Medium and Low severity issues.