BAIL security

Euler.Finance
EUL ERC20

FINAL REPORT

June '2025

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	Euler.Finance - EUL ERC20
Website	euler.finance
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/euler-xyz/euler-governance/blob/1249ea8be20cd10b795251ba18bc785a26c1361f/contracts/governance/Eul.sol

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)	Failed Resolution
High					
Medium					
Low					
Informational	4			4	
Governance					
Total	4			4	

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium-level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless, the issue should be fixed immediately.
Informational	Effects are small and do not pose an immediate danger to the project or users.
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior.

3. Detection

EUL

The EUL contract is a simple ERC20 contract extended with ERC20Votes which allows for the delegation of voting power and permit approvals. It furthermore leverages OpenZeppelin's AccessControl contract for RBAC.

The contract implements a customized minting mechanism which allows any address with the ADMIN_ROLE to call the mint function whenever MINT_MIN_INTERVAL (365 days) since the last mint has passed. For the very first mint, this will be after the customized mintingRestrictedBefore has surpassed, which is at TS = 1767225600.

The contract will then mint MINT_MAX_PERCENT (2.718%) from the current totalSupply to the treasury:

> totalSupply * 2718 / 100000

Simplified this means the maximum inflation of the token can be 2.718% / year, *after* mintingRestrictedBefore has surpassed. The very first mint (after the initial mint) can thus happen earliest at Thu Jan 01 2026 00:00:00 GMT+0000.

All inherited files originate from OpenZeppelin (ERC20Votes and AccessControl) and are not modified. *These files are not explicitly part of the audit scope - it is assumed that these are bug-free*.

Appendix: ERC20 Votes

ERC20Votes is an OpenZeppelin extension that adds governance voting capabilities to a standard ERC20 token.
It keeps a historical record of each account's voting power and makes that power available for on-chain governance processes.
Users can delegate their votes to themselves or another address, which allows flexible representation in governance decisions.
Voting units typically map 1:1 to token balances, but only delegated balances actively count in voting outcomes.
The module limits total token supply to type(uint208).max by default, preventing overflows and ensuring accurate checkpoint math.

Deployed Blockchain:

Ethereum

Deployed Address:

https://etherscan.io/token/0xd9fcd98c322942075a5c3860693e9f4f03aae07b

Trust Assumptions:

Any role with the ADMIN_ROLE can call mint and updateTreasury

Decimals:

18

Initial mint:

27,182,818

Minting occasions:

Minting happens during deployment and via the mint function. The mint function is restricted to be called once a year.

Core Invariants:

INV 1: Minting can only happen once mintingRestrictedBefore is surpassed

INV 2: Each mint can only mint 2.718% (e) of the totalSupply

INV 3: Only addresses with the ADMIN_ROLE can mint and change the treasury address

Privileged Functions

- mint
- updateTreasury
- grantRole
- revokeRole

Issue_01	Change of treasury will not automatically grant privileged roles to new treasury
Severity	**Informational**
Description	Currently, any address with the ADMIN_ROLE can change the treasury address. At the same time, the treasury address holds the ADMIN_ROLE (see constructor). A change of the treasury address however does not automatically grant the new treasury address the said privilege.
Recommendations	Consider manually granting the new treasury address this privilege.
Comments / Resolution	Acknowledged, the contract is already deployed and the client will follow the aforementioned recommendation.

Issue_02	Lack of validation for mintingRestrictedBefore to be not too far in the future
Severity	**Informational**
Description	Within the constructor, the mintingRestrictedBefore value can be any timestamp as long as it is past the current block.timestamp. This leaves room for errors in the scenario where for example a typo is provided in the timestamp or an additional zero/digit. It would effectively block minting.
Recommendations	Consider validating the mintingRestrictedBefore_ parameter accordingly.
Comments / Resolution	Acknowledged, the contract is already deployed.

Issue_03	Lack of event emission
Severity	**Informational**
Description	Currently, the contract does not expose events for the mint function. Sensitive functions that change the contract state should emit events to make it easier for users to identify the contract state.
Recommendations	Consider emitting an event whenever the mint function is called.
Comments / Resolution	Acknowledged, the contract is already deployed.

Issue_04	Lack of accounting for leap years
Severity	**Informational**
Description	Currently, minting is allowed every 365 days. This is most likely attached to “once per year”. However, it is technically not 100% correct as leap years are not accounted for.
Recommendations	Consider accounting for leap years.
Comments / Resolution	Acknowledged, the contract is already deployed.