

(<https://blog.coinfabrik.com/>)

(<https://blog.coinfabrik.com/>)

SMART CONTRACT AUDIT ([HTTPS://BLOG.COINFABRIK.COM/CATEGORY/SMART-CONTRACTS/SMART-CONTRACT-AUDIT-SMART-CONTRACTS/](https://blog.coinfabrik.com/category/smart-contracts/smart-contract-audit-smart-contracts/))

# EMDX – Token Contracts Audit

Heloisa Ceni (<https://blog.coinfabrik.com/author/heloisa-ceni/>)

December 2, 2021 (<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/emdx-token-contracts-audit/>)



## Contents



1. Introduction
2. Summary
3. Contracts
4. Analyses
5. Findings and Fixes
6. Severity Classification
7. Issues Found by Severity
  - 7.1. Critical Severity
  - 7.2. Medium Severity
    - 7.2.1. ME-01 Denial of Service in Vesting.updateScore()
    - 7.2.2. ME-02 Possible Overflow in Vesting.updateScore()
  - 7.3. Minor Severity
  - 7.4. Enhancements
    - 7.4.1. EN-01 SafeMath Libraries Removal
8. Other considerations
  - 8.1. Vesting schedule
9. Conclusion

## Introduction

CoinFabrik was asked to audit the contracts for the EMDX project. First, we will provide a summary of our discoveries and then we will show the details of our findings.

## Summary

The contracts audited are from the EMDX repository at <https://github.com/emdx-dex/token-contracts.git>. The audit is based on the commit 2a376848d3d0a3604f8c6ca702fe4718019d1b3d. Fixes were made and rechecked based on the commit 2ffe6fa5eba3759275a65f855ae26e5954f4cd6c.

# Contracts

The audited contracts are:

- contracts/EMDXToken.sol:SimpleERC20token
- contracts/Vesting.sol:Vestingcontract

# Analyses

The following analyses were performed:

- Misuse of the different call methods
- Integer overflow errors
- Division by zero errors
- Outdated version of Solidity compiler
- Front running attacks
- Reentrancy attacks
- Misuse of block timestamps
- Softlock denial of service attacks
- Functions with excessive gas cost
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Failure to use a withdrawal pattern
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures

# Findings and Fixes

ID	Title	Severity	Status
ME-01	Denial of Service in Vesting.updateScore()	Medium	Mitigated
ME-02	Possible Overflow in Vesting.updateScore()	Medium	Fixed
EN-01	SafeMath Libraries Removal	Enhancement	Acknowledged

# Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues

do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

- **Enhancement:** These kinds of findings do not represent a security risk. They are best practices that we suggest implementing. This classification is summarized in the following table:

SEVERITY	EXPLOITABLE	ROADBLOCK	TO BE FIXED
Critical	Yes	Yes	Immediately
Medium	In the near future	Yes	As soon as possible
Minor	Unlikely	No	Eventually
Enhancement	No	No	Eventually

## Issues Found by Severity

### Critical Severity

No issues found.

### Medium Severity

#### ME-01 Denial of Service in `Vesting.updateScore()`

The update score function will not work if any of the transfers in line 172 fails, reverting the transaction.

##### Recommendation

Use the “Favor pull over push for external calls” pattern to solve this problem. See <https://eth.wiki/howto/smart-contract-safety#favor-pull-over-push-for-external-calls> for details.

##### Solution

The development team informed us that they decided that the “Favor pull over push for external calls” pattern is not required in this instance because:

- less than 10 transfers will be made on each call.
- all the funds required will always be available.
- the token to be used is EMDXToken (in the same git repository and also part of this audit), making it not possible to trigger the denial of service.

#### ME-02 Possible Overflow in `Vesting.updateScore()`

An overflow may occur if `totalAmount` is bigger than  $2^{256}/99$ . This overflow will block the `Vesting.updateScore()` function on every invocation, stopping the recovery of any token owned by the Vesting contract with a smaller or similar score to the one passed in the `_newScore` variable.

Hereunder are the lines 157-165 of Vesting.sol. The expression where the overflow will be triggered is `_newScore.mul(remainingAmount)`

```
LockVesting memory lock = locks[beneficiaries[i]];
// calculate already vested percentage
uint256 remainingAmount = lock.totalAmount.sub(
    lock.releasedAmount
);
// calculate amount to be vested
uint256 releasableAmount = _newScore.mul(remainingAmount).div( 100
);
```

### Recommendation

Add a required statement in the `Vesting.grantVesting()` function checking that the `_amount` does not exceed  $2^{256}/99$ .

### Solution

The issue was fixed by the development team.

## Minor Severity

No issues found.

## Enhancements

### EN-01 SafeMath Libraries Removal

Given that the solidity version is  $\geq 0.8$  there is no need to use SafeMath libraries. Usage of “@openzeppelin/contracts/utils/math/SafeMath.sol” in Vesting.sol is not required. See <https://docs.soliditylang.org/en/v0.8.9/080-breaking-changes.html#how-to-update-your-code> for details (section “How to update your code”).

### Solution

The development team informed us that they decided not to do this enhancement because they want to minimize the code changes to be made.

## Other considerations

### Vesting schedule

While in standard vesting schedules the amount that can be vested on each period is decided when the vesting is started, in the Vesting contract the tokens awarded on each period are decided by a score passed by the oracle using the `Vesting.updateScore()` function in lines 131-185 of Vesting.sol, when each vesting period starts. This includes the option to set the score as 0 or not calling the `Vesting.updateScore()`, which would not award any tokens for the period.

## Conclusion

We found 2 medium-severity issues. Also, an enhancement was recommended.

One of the medium-severity issues was fixed and the other was mitigated. The enhancement was not considered necessary by the development team.

*Disclaimer: This audit report is not a security warranty, investment advice, or approval of the EMDX project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.*

## Related Posts

(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/stasis-token-smart-contract-audit/>)  
Stasis Token Smart Contract Audit  
(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/stasis-token-smart-contract-audit/>)  
Coinfabrik has been hired to audit the smart contracts for Stasis sale, the Stable Euro...

(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/>)  
DMToken Token Sale Smart Contract Audit  
(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/>)  
Coinfabrik was hired to audit the contract in terms of its security. First of all,...

(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bricks4us-token-smart-contract-security-audit/>)

Bricks4Us was asked to audit the contracts for the Bricks4Us token sale. Firstly, we will...

(<https://blog.coinfabrik.com/smart-contracts/etherparty-token-smart-contract-security-audit-coinfabrik/>)

EtherParty Smart Contract Security Audit

Coinfabrik team has been hired to audit Etherparty smart contracts. Firstly, we will provide a...

Tags:

audit  
(<https://blog.coinfabrik.com/tag/audit/>)


auditoria  
(<https://blog.coinfabrik.com/tag/auditoria/>)


auditoria de seguridad  
(<https://blog.coinfabrik.com/tag/auditoria-de-seguridad/>)


security audit  
(<https://blog.coinfabrik.com/tag/security-audit/>)


Token  
(<https://blog.coinfabrik.com/tag/token/>)

SHARE  
ON

 (<https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/emdx-token-contracts-audit/>)

 (<https://twitter.com/intent/tweet?text=EMDX%20Token%20Contracts%20Audit&url=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/emdx-token-contracts-audit/>)

 ([https://pinterest.com/pin/create/button/?url=https://blog.coinfabrik.com/wp-content/uploads/2021/12/EMDX\\_logo\\_bg.png&description=EMDX+%26%238211%3B+Token+Contracts+Audit&media=https://blog.coinfabrik.com/wp-content/uploads/2021/12/EMDX\\_logo\\_bg.png](https://pinterest.com/pin/create/button/?url=https://blog.coinfabrik.com/wp-content/uploads/2021/12/EMDX_logo_bg.png&description=EMDX+%26%238211%3B+Token+Contracts+Audit&media=https://blog.coinfabrik.com/wp-content/uploads/2021/12/EMDX_logo_bg.png))

 (<https://www.linkedin.com/shareArticle?mini=true&url=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/emdx-token-contracts-audit/&title=EMDX%20Token%20Contracts%20Audit&source=CoinFabrik%20Blog>)

**Bit2Me Token – Security Audit**  
(<https://blog.coinfabrik.com/smart-contract-audit/bit2me-token-security-audit/>)

**AlexGo Audit**  
(<https://blog.coinfabrik.com/smart-contracts/alexgo-audit/>)

## You may also like

(<https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/>)

**Magic Bridge Audit** (<https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/>)

(<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablelegatedmarketplace/>)

2 months ago

Smart Contract Audit (<https://blog.coinfabrik.com/category/smart-contracts/smart-contract-audit-smart-contracts/>)

**MintingFactoryV2, BaseUpgradableMarketplace & KODAV3UpgradableGatedMarketplace** (<https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablelegatedmarketplace/>)

 (<https://blog.coinfabrik.com/feed/>)

 (<https://ar.linkedin.com/company/coinfabrik>)

 (<https://twitter.com/coinfabrik>)

  
(<https://www.youtube.com/channel/UC2GmjCr7aEz-il31kqOy9aw>)

 (<https://www.facebook.com/CoinFabrik/>)

 (<https://www.reddit.com/r/CoinFabrik/>)

 (<https://github.com/coinfabrik>)

---

© 2021 CoinFabrik - All Rights Reserved.

Made with ❤ in Buenos Aires, Argentina.