Moeda Token Audit

OPENZEPPELIN SECURITY | MAY 16, 2017

Security Audits

The Moeda team asked us to review and audit their new Moeda Token code. We looked at their contracts and now publish our results.

The audited contracts can be found in their moeda repo. The version used for this report is commit **b2bf23119d563e251b6f16b29b642bac43e76a64**. The main contracts are MoedaToken.sol and Crowdsale.sol.

Overall the code is good and has only minor issues. Here's our assessment and recommendations, in order of importance.

Update: The Moeda team implemented most of our recommendations in the latest version of their code.

Severe

We haven't found any severe security problems with the code.

Potential problems

Unnecessary complexity calculating token amount

The formula used to calculate the token amount in line 116 is unnecessarily complicated. It could be replaced by a multiplication by the amount of tokens bought with one wei for each given tier. Consider making this change by replacing the token creation rates by `1 ether` divided by the current values. This has a number of benefits:

3. It's simpler, which is always preferable. When code is irreducibly complex, consider using temporary variables to improve readability.

Update: Fixed in https://github.com/erkmos/moeda/commit/a894b2cccef0236e90326b84a0b3dde485e78cd1.

Conditions for finalization

The crowdsale can be finalized at any moment, in particular before `endBlock` is reached. Please double check that this is the desired behavior.

Update: Now requiring that either cap or end block has been reached. Fixed in https://github.com/erkmos/moeda/commit/5eb52c0648d1af0db11f5eb883379d6c8054e2c0 and refactored in https://github.com/erkmos/moeda/commit/9edf6fe6e23aec7f6fcba6d0921436e435336048.

Use of Transfer event to log token creation

The crowdsale contract uses the Transfer event to log the creation of tokens, with a *from* field set to the zero address. This is overloading the semantics of the `Transfer` event of the ERC20 standard, which might not be a good idea as other software (such as wallets or exchanges) could be relying on these semantics. This is an ongoing discussion in the ERC20 standardization process, but our stance on it is that a different event should be used.

Update: Created a new event, fixed in https://github.com/erkmos/moeda/commit/68ff6544976c2bfa655cae28e69ab09d813dbc46.

Warnings

Use latest version of Solidity

The contracts are written for an old version of the Solidity compiler (0.4.8). We recommend changing the solidity version pragma to the latest version (`pragma solidity ^0.4.11;`) to enforce the use of an up to date compiler. (Please note that there has been a recent update to the Truffle framework to make it compatible with 0.4.11.)

Unexpected throw

When the ether received adds up to exactly `TIER3_CAP`, `getLimitAndRate` throws when the comment says it shouldn't. Consider changing all the comparisons in this function to inclusive ones (e.g. `totalReceived <= TIER3_CAP`), which is the standard behavior in these cases.

Redundant return value

`processBuy` function in line 126 is declared to return a boolean but never returns `false`. Instead, it always throws in case of error. Consider sticking to one error signaling convention. We recommend `throw`ing to fail as early and loudly as possible, ensuring the propagation of the error. Same thing with unlock function in line 40 of MoedaToken.sol.

Enforce all preconditions

`processBuy` function in line 126 has a precondition written in the comments which is not enforced in the code: it's only usable while the crowdsale is active. In the current state of the contract, the function is internal and only called from the fallback function, which is properly guarded with the `onlyDuringSale` modifier. However, it's best to enforce preconditions independently on a function-by-function basis.

Consider using the `onlyDuringSale` modifier at the beginning of `processBuy`. Furthermore, since `processBuy` is only used once, and is the only thing the fallback function does, consider removing it altogether, moving the code inside the fallback function.

Update: The function has merged with the fallback function, was fixed in https://github.com/erkmos/moeda/commit/d4c8f85809ecd33852924aa71b0a233f4dbc2318

Naming suggestions

- The boolean `locked` variable in line 16 of MoedaToken.sol is not descriptive of its actual use. Consider a name like `saleActive`.
  Update: Was renamed to saleActive in https://github.com/erkmos/moeda/commit/8e9b6861d7b735c1ef49c7594dfc5e0cc777fbc6

Update: This has been renamed to PRESALE_TOKEN_ALLOCATION (it is actually referring to an allocation for a previous sale), fixed in https://github.com/erkmos/moeda/commit/7ed3d1d78d8f8da7705e0b93351659f058860a4c.

- The word “rate” is ambiguous (lines 28–30). consider calling it “price”. Bear in mind that if the changes suggested in “Unnecessary complexity calculating token amount” are implemented, the name “price” would no longer be correct.
- The `Buy` event in line 37 of Crowdsale.sol could have a better name. In general, events have nouns or past-tense verbs as names. Consider changing it to `Purchase` or `TokenBought`.
  Update: Event renamed to Purchase in https://github.com/erkmos/moeda/commit/4858c60cef148bbbdaed87dbc3965f58a0787ab5.

Use SafeMath in all math operations

Most math operations are safe, except a sum in the token contract at line 51. It’s always better to be safe and perform checked operations.
Update: The place it was missing has been fixed in https://github.com/erkmos/moeda/commit/5e1ccfb7b4f934c0588c84196e435e21f46c7de5.

Additional Information and Notes

- There are MLO tokens assigned to the `wallet` address in line 152 of Crowdsale.sol. Please make sure the wallet contract is capable of holding and managing tokens.
  Update: included the wallet contract in https://github.com/erkmos/moeda/commit/86cf3a632b73ba8993c1477b103f760a13a785fc
- Contracts use “ether” units to specify token amounts (see Crowdsale.sol, line 22, MoedaToken.sol, line 13). This is okay but slightly confusing and you should be careful.
  Update: Changed to named constants in https://github.com/erkmos/moeda/commit/ef08e24c44a0f6144d6e41d77f5daae4f6272928.
- Good job validating preconditions for the constructor parameters.
- `presaleWallet` in line 19 of Crowdsale.sol is never used and should be removed.
  Update: Was removed in https://github.com/erkmos/moeda/commit/171959dda8ed7a3ff10d80bbf65044298d06a46d.

The fallback function is redundant since it only calls the [illegible] function.

Update: Was fixed in https://github.com/erkmos/moeda/commit/d4c8f85809ecd33852924aa71b0a233f4dbc2318.

- The `totalTokensSold` variable is only written to, and never read. Moreover, it's redundant because the information is already available in the `totalSupply` state variable of the token contract. Consider removing it; redundancy can be a source of problems.
- A call of `send` isn't the last expression in the function (line 129 of Crowdsale.sol). We haven't found any problem with this, but it has been a source of issues in the past. Consider moving the `send` call to the end of the function.

  Update: fixed in https://github.com/erkmos/moeda/commit/5a5ec164e6ccb601f4da19d897522e8ca3bcfc01 (along with some more redundant booleans being removed).
- All tests run correctly and pass. ✓

Conclusions

No severe security issues were found. Some additional changes were proposed to follow best practices and reduce potential attack surface.

Update: The Moeda team implemented most of our recommendations in the latest version of their code.

Overall the code is good and has only minor issues.

Thanks to Francisco Giordano for helping write this report.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the Moeda Token contract. We have not reviewed the related Moeda project. The above should not be construed as investment advice or an offering of tokens. For general information about smart contract security, check out our thoughts here.

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits