Maple

Smart Contract Preliminary Security Audit

Date 29/May/2021

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and ApeAudits and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (ApeAudits) owe no duty of care towards you or any other person, nor does ApeAudits make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and ApeAudits hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, ApeAudits hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against ApeAudits, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Purpose

This document details ApeAudits' findings and recommended solutions. It is the first step in our relationship with you, the client. Please read carefully, fix any issues, and report back to us with the revised contracts and any concerns you may have. This document will not be displayed to the public, and as such does not contain any summarized information. The final, public-facing report will contain more information detailing the resolution of all issues, as well as summarized + categorized information about all issues.

Maple.sol

High Severity Issues

- **Issue:**
  If lock() is called before renounceOwnership(), unlock() allows ownership to be reclaimed after renouncing.

  Recommendation:
  Remove lock() and unlock() (recommended) OR unset _previousOwner in unlock().

- **Issue:**
  Line 1060, in _sendToSS(uint) - incorrectly adding reflect value to true value.

  Recommendation:
  This code:
  ```
  _tOwned[WelfareCommandCenterAddress] =
  _tOwned[WelfareCommandCenterAddress].add(rSS);
  ```

  Should be changed to:
  ```
  _tOwned[WelfareCommandCenterAddress] =
  _tOwned[WelfareCommandCenterAddress].add(tSS);
  ```

- **Issue:**
  EmergencyAddress is never set, blocking the use of any function using the onlyEmergency modifier.

  Recommendation:
  Set EmergencyAddress or use a different modifier for the affected functions.

Medium Severity Issues

- **Issue:**
  updateWelfareCommandCenter(address payable), updateSocialSecurity(address), and updateBonusVault(address) can be used more than once.

  Recommendation:
  We recommend limiting these functions to only one use. However, it is apparent that these functions are meant for initialization and emergency purposes. If they are to

remain callable, protect the administrative keys extremely carefully.

- **Issue:**
  In the function _transferToSS(address, address, uint256):
  1. Use of _tOwned assumes the sender and receiver are both excluded from reflection rewards.
  2. Adding true values (tAmount) to reflect balance (_rOwned).

  Recommendation:
  From the structure of the other contracts, it seems that this function is no longer meant to be used. Please remove it if it isn't going to be used.

- **Issue:**
  Fees add up to 30% in total, which could make the token difficult to buy at launch time.

  Recommendation:
  This seems to be intentional, but consider lowering fees at launch time and raising them after.

Low Severity Issues

- **Issue:**
  Unused variables - budgetSwitch, liquiditySubmitted.

  Recommendation:
  Remove unused variables.

- **Issue:**
  Potentially usable code commented out in constructor and _getTValues(uint256).

  Recommendation:
  Either remove or uncomment the code.

- **Issue:**
  Unclear function name: _tradeCheck(uint256).

  Recommendation:
  Choose a name that makes reference to the triple liquidity tax.

MapleVault.sol

High Severity Issues

- **Issue:**
  Line 484, in ssVaultDeposit(uint256, uint256) - SSVault expiration time is fixed at 30 days after deposit time and does not vary based on staking time.

  Recommendation:
  Multiply timePeriod by _timeLockUnits.

- **Issue:**
  Line 496, in ssVaultDeposit(uint256, uint256) - _depositamount carries the BonusVault bonus, which hasn't been approved by the user and also has already been deposited.

  Recommendation:
  In applyBonus(uint256, uint256), return only the bonus. In the calling code, deal with _depositamount and the bonus separately.

- **Issue:**
  applyBonus(uint256, uint256) is public and does not perform any validation of _timeLockUnits or _depositamount.

  Recommendation:
  Make applyBonus private.

- **Issue:**
  settle(uint256) can be called multiple times, calling deductGlobalDeposits more than once, which could allow an attacker to manipulate globalDepositTimeValue.

  Recommendation:
  Make sure it can only be called once.

- **Issue:**
  Line 604, in settle(uint256) - the transfer should take place before computing the penalty, and should transfer from the SSVault, not msg.sender.

  Recommendation:
  Move this line to before line 602 and change msg.sender to _ssvaultaddress.

- **Issue:**
  deductGlobalDeposits(uint256, uint256) has onlySSVault modifier, but is not called from an SSVault.

  Recommendation:
  Remove the modifier.

- **Issue:**
  Line 539, in depositSSTax() - lastUpdateTime is never updated, which enables DoS.

  Recommendation:
  Update lastUpdateTime inside depositSSTax at least.

- **Issue:**
  EmergencyAddress is never set, blocking the use of any function using the onlyEmergency modifier.

  Recommendation:
  Set EmergencyAddress or use a different modifier for the affected functions.

Medium Severity Issues

- **Issue:**
  updateWelfareCommandCenter(address), updateWelfareAddress(address), and updateBonusVault(address) can be used more than once.

  Recommendation:
  We recommend limiting these functions to only one use. However, it is apparent that these functions are meant for initialization and emergency purposes. If they are to remain callable, protect the administrative keys extremely carefully.

- **Issue:**
  Line 520, in pullTaxShare(uint256, uint256) - the accumulated total tax is added to globalSSTaxDepositsCollectedByPensioners[x] instead of the available tax at timestep x.

  Recommendation:
  Consider calculating the tax share for timestep x and adding it to globalSSTaxDepositsCollectedByPensioners[x] instead of the total.

Low Severity Issues

- **Issue:**
  In SocialSecurity constructor - redundant calls to getBonusVault() and getWelfareAddress() since WelfareCommandCenter::initialize() does the same thing.

  Recommendation:
  Remove redundant code.

- **Issue:**
  timeValueMultiplier(uint256) uses a loop, but can be exchanged for a mathematical function.

  Recommendation:
  This function is equivalent to _timeLockUnits + (_timeLockUnits * (_timeLockUnits - 1) / 2).

- **Issue:**
  Misleading error messages in the modifiers onlySSVault(), OnlyEmergency(), and OnlyCommandCenter().

  Recommendation:
  Choose error messages that are more accurate.

Bonus.sol

High Severity Issues

- No high severity issues found.

Medium Severity Issues

- **Issue:**
  updateCommandCenter(address), updateWelfareAddress(address), updateSSAddress(address), and updateBonusVaultAddress() can be used more than once.

  Recommendation:
  We recommend limiting these functions to only one use. However, it is apparent that these functions are meant for initialization and emergency purposes. If they are to remain callable, protect the administrative keys extremely carefully.

- **Issue:**
  Line 349, in approve() - the approved amount may be lower than desired.

  Recommendation:
  Consider using ~uint256(0) for the approval amount.

Low Severity Issues

- **Issue:**
  Line 306, in interface Welfare - withdrawBNB() is not used or part of the Welfare interface.

  Recommendation:
  Remove this line.

MapleCenter.sol

High Severity Issues

- No high severity issues found.

Medium Severity Issues

- **Issue:**
  updateWelfareAddress(address), updateSSAddress(address), and updateBonusVaultAddress(address) can be used more than once.

  Recommendation:
  We recommend limiting these functions to only one use. However, it is apparent that these functions are meant for initialization and emergency purposes. If they are to remain callable, protect the administrative keys extremely carefully.

- **Issue:**
  Line 363, in approve() - the approved amount may be lower than desired.

  Recommendation:
  Consider using ~uint256(0) for the approval amount.

- **Issue:**
  Line 328 - lever is never updated.

  Recommendation:
  Either add some updating functionality or remove the variable altogether.

Low Severity Issues

- **Issue:**
  Line 306, in interface Welfare - withdrawBNB() is not used or part of the Welfare interface.

  Recommendation:
  Remove this line.

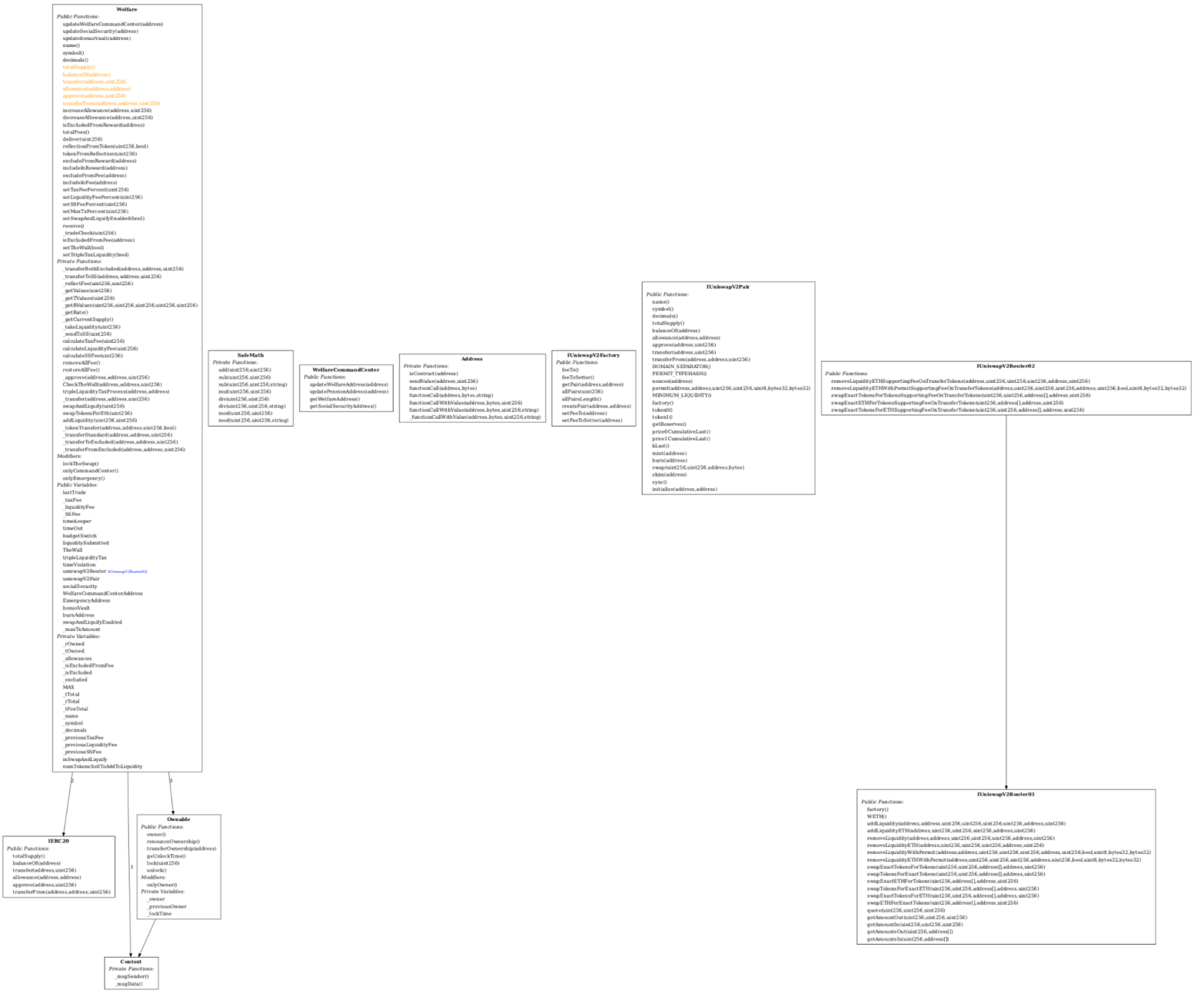
Conclusion

Many high severity issues found in Maple.sol and MapleVault.sol. We strongly recommend writing automated tests and performing thorough testing on a testnet and/or local environment and re-evaluating with ApeAudits before deploying to mainnet. Our first re-evaluation comes at no additional charge to you.

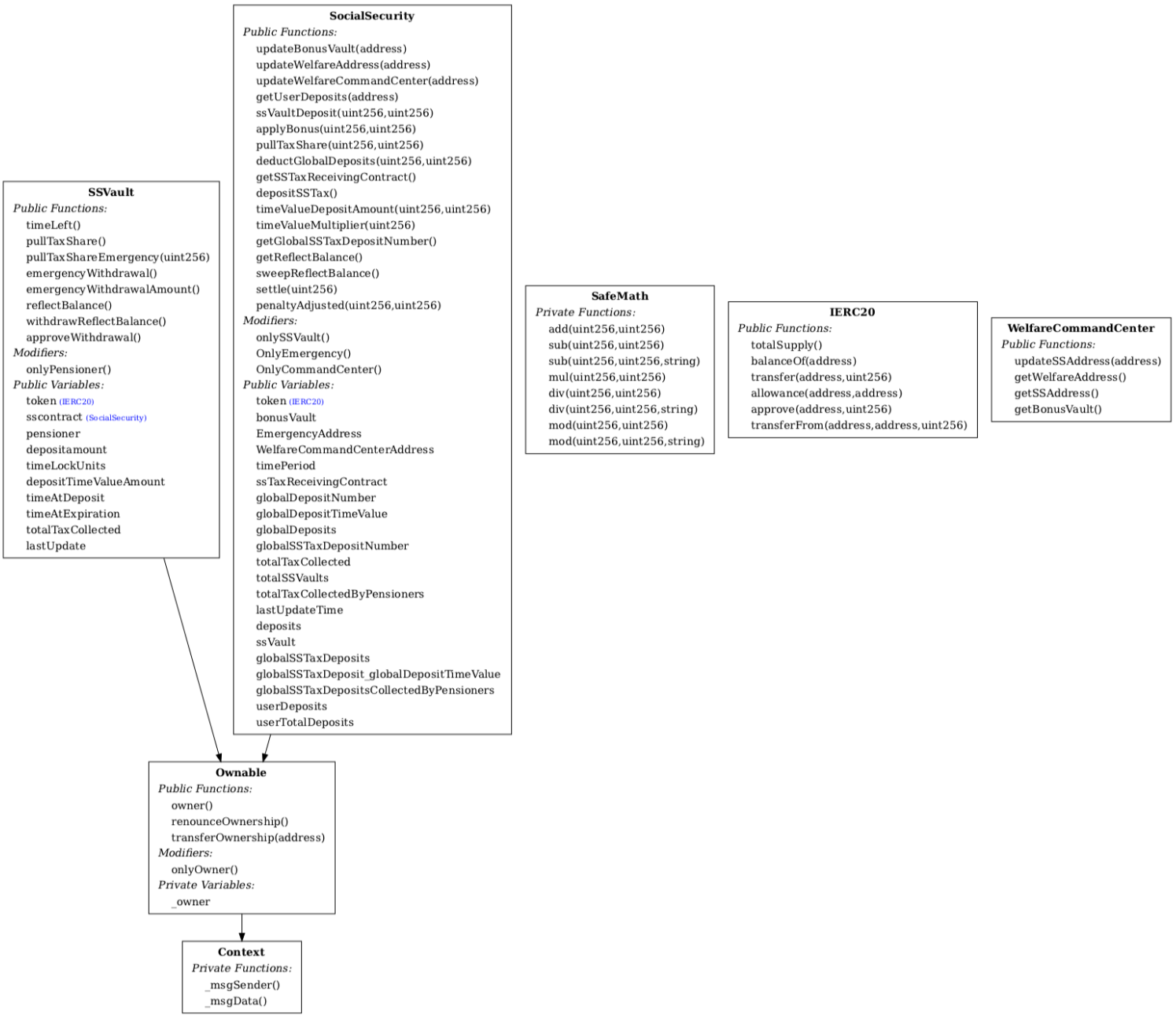
ApeAudits note:
Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Appendix

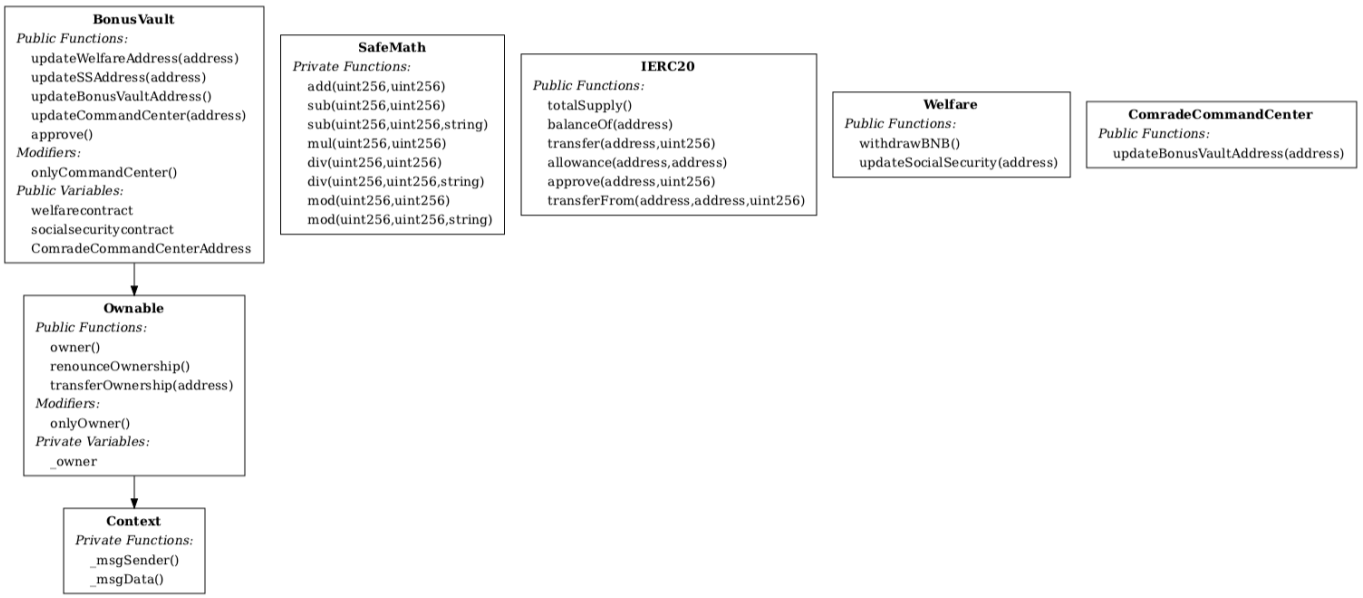
Maple.sol:



MapleVault.sol:



Bonus.sol:



MapleCenter.sol: