LatteSwap v2 Security Audit

October 4, 2021

WATCHPUG

Table of Contents

Summary

This report has been prepared for **LatteSwap v2** smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

Scope

Our review focused on the upgraded LATTEv2 token contract `LATTEV2.sol`, `BeanBagV2.sol` and `MasterBarista.sol`.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**LatteSwap v2**
Codebase	**https://github.com/latteswap-official/latteswap-contract**
Commit	**8c9338d45255f9e134805999c52b1f25acd0570f**
Language	**Solidity**
Platform	**BSC**

Audit Summary

Delivery Date	**Oct 4, 2021**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**4**

LS-1: MasterBarista.sol LATTEv2 token should not be allowed in deposit() and withdraw()

Medium

Issue Description

contracts/farm/MasterBarista.sol

```
function deposit(
  address _for,
  address _stakeToken,
  uint256 _amount
) external override onlyPermittedTokenFunder(_for, _stakeToken) nonReentrant {
  _assignActiveToken();
  require(
    _stakeToken != address(0) && _stakeToken != address(1),
    "MasterBarista::setPool::_stakeToken must not be address(0) or address(1)"
  );
  require(_stakeToken != address(latte), "MasterBarista::deposit::use depositLatte instead");
  // MISSING CHECK IF _stakeToken != address(latteV2)
  require(pools.has(_stakeToken), "MasterBarista::deposit::no pool");
  ...
```

`LATTEv2` should not be allowed in `deposit()` and `withdraw() `as BEAN token is used and required when withdrawing LATTE token.

Recommendation

Consider adding the missing check for `_stakeToken != address(latteV2)` in `deposit()` and `withdraw()`.

Status

✓ Fixed in commit: c3788d2a9a96823a2ca4e0a937326e3492c55def.

LS-2: LATTEv2.sol#claimLock() can be simplified

Issue Description

contracts/farm/LATTEV2.sol

Recommendation

With a snapshot taken, the locked amounts are fixed and the LATTEv2 token intends not to adopt any changes to the locked amounts in v1 (changed by `transferAll`).

Therefore, the `claimLock()` function and related functions (`_setClaimed`, `isClaimed`) and variables (`merkleRoot`, `claimedBitMap`) can be simplified and removed by changing it into a push style airdrop-like function.

The integrity of the data can be checked by comparing the `totalLocked` amount in LATTEv2 and LATTEv1.

Furthermore, with this change being made, there will be no action required for users anymore.

Resolution

The recommended changes have been made in commit a8ded1de765a39a4072beb520d0bf90f57aa8e7e.

LS-3: MasterBarista.sol#_harvest() Add check if bonus > 0 can save gas

Informational

Issue Description

contracts/farm/MasterBarista.sol

```
  function _harvest(
    address _for,
    address _stakeToken,
    uint256 _lastRewardBlock
  ) internal {
    _assignActiveToken();
    PoolInfo memory pool = poolInfo[_stakeToken];
    UserInfo memory user = userInfo[_stakeToken][_for];
    require(
      user.fundedBy == _msgSender() || _msgSender() == 0xE626f...283,
      "MasterBarista::_harvest::only funder"
    );
    require(user.amount > 0, "MasterBarista::_harvest::nothing to harvest");
    uint256 pending = user.amount.mul(pool.accLattePerShare).div(1e12).sub(user.rewardDebt);
    require(
      pending <= activeLatte.balanceOf(address(activeBean)),
      "MasterBarista::_harvest::wait what.. not enough LATTE"
    );
    uint256 bonus =
user.amount.mul(pool.accLattePerShareTilBonusEnd).div(1e12).sub(user.bonusDebt);
    activeBean.safeLatteTransfer(_for, pending);
    if (stakeTokenCallerContracts[_stakeToken].has(_msgSender())) {
      _masterBaristaCallee(_msgSender(), _stakeToken, _for, pending, _lastRewardBlock);
    }
    activeLatte.lock(_for, bonus.mul(bonusLockUpBps).div(10000));

    emit Harvest(_msgSender(), _for, _stakeToken, pending);
  }
```

Every call to an external contract costs a decent amount of gas.

After the migration, `bonus` should be 0 for most users until a new bonus period starts, check if `bonus > 0` before calling `activeLatte.lock` can save gas.

Status

✓ Fixed in commit: 6d4fc157bb4d41f5c5a84edbb44ab3379d9d80f7.

LS-4: MasterBarista.sol Unused code

Informational

Issue Description

contracts/farm/MasterBarista.sol

```
modifier onlyPermittedTokensFunder(address _beneficiary, address[] calldata _stakeTokens) {
  for (uint256 i = 0; i < _stakeTokens.length; i++) {
    require(
      _isFunder(_beneficiary, _stakeTokens[i]),
      "MasterBarista::onlyPermittedTokensFunder: caller is not permitted"
    );
  }
  _
```

The modifier `onlyPermittedTokensFunder` is unused.

Recommendation

Consider removing unused code.

Status

✓ Fixed in commit: bfa04a60c43f318262ba95f67a25c290cecc9fcb.

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, 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, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.