GEMZ
SECURITY
REPORT
PREPARED ON
3/26/2021

BLIZZARD.MONEY

Code Review

Table of Contents

Report

Included below is a code security audit of the Blizzard.Money project. The information appearing in this report is for general and educational purposes only and is not intended to provide any legal security guarantees to any individual or entity. It is advisable to conduct more reviews or/and audits as just one review or audit may miss important issues.

This report does not provide personalized investment advice or recommendations, nor does it provide advice to perform any transactions and it does not provide investment, financial, legal, or tax advice. We are not responsible or liable for any loss which results from the report. This report should not be considered as investment advice.

Farm
Blizzard Token (BLZD)

Address: 0x57067a6bd75c0e95a6a5f158455926e43e79beb0

The Blizzard token is the reward/governance for Blizzard.Money. Users receive this for farming in liquidity pools and staking in specific pools.

The owner of BlzdToken is 0x2078f4a75c92a6918d13e3e2f14183443ebf55d3 (MasterChefV2), and not an externally owned address (EOA). This ownership restricts the minting of more Blzd tokens to only MasterChefV2.

MasterChefV2
Address: 0x2078f4a75c92a6918d13e3e2f14183443ebf55d3

Owner
The owner of the MasterChefV2 contract is 0x82fdd95b4049c94f209ba68d5cb157cb3f2174da, which is a Timelock contract.

The owner of this contract is able to call the following owner-only functions:

- add
- set
- renounceOwnership
- transferOwnership

Tokenomics

The specified emission per block is 1 BLZD, meaning that 1 BLZD is created per block on BSC. This value has been initialized in the constructor upon contract creation and there is no code with functionality to change this value.

BONUS_MULTIPLIER is hardcoded as a constant of 1, so it is not possible to change the rate of emission by modification of this multiplier using this value. In short, the emissions of BLZD will remain at 1 BLZD in perpetuity.

108.8% of the block emission is minted, with 100% (1) as the block rewards, and 8.8% (0.088) going to the dev address. The owner of the dev address has full discretion over the use of these funds.

As there is no cap to the supply of BLZD; it is an inflationary token.

Deposits and withdrawals

Farming pools are subject to deposit fees, in this case, up to 4%. This deposit fee is set at the time of adding a pool, and can be modified from any value between 0% to 4%. The maximum cap of the deposit fee is 4%.

When depositing into a pool with a deposit fee, the fee percentage will be subtracted from the deposit amount.

This fee will be divided by half and sent to 2 external addresses, feeAddBb and feeAddSt. The remaining amount after fee deduction will be added as the user's deposited amount into the pool.

The nonReentrant modifier is used for deposit, withdraw, and emergencyWithdraw.

Others
There is no migrator or similar migration function to transfer user deposited tokens from the contract to another address. In the past, the migrator function has been used to withdraw user funds or “rug” other projects on BSC.

Timelock
Address: 0x82fdd95b4049c94f209ba68d5cb157cb3f2174da
The timelock uses Compound Finance’s timelock contract, with the minimum delay set to 12 hours. At the time of this review, it was verified that the timelock is set to 24 hours delay (86400 seconds).

Issues

Rated on a scale of: Infomational, Low, Medium, High, Critical

BLZ-001: massUpdatePools() can be optional when modifying pool weightage

Severity: Infomational

In addPool and setPool, the owner can decide whether massUpdatePools() will be called by passing _withUpdate as true. This can affect other existing pools' distribution of yield if false is passed, causing existing pools to not be updated before a change to the pool weights are done.

Recommendations:
If _allocPoint is greater than 0, massUpdatePools() should be mandatory. The if statement can be change to the following
if (_withUpdate || _allocPoint > 0)

Alternatively, the owner needs to always have _withUpdate set to true in the following cases:
- If _allocPoint is non 0 for add
- Every call of set

BLZ-002: Dust in MasterChefV2 due to division in fee calculation

Severity: Infomational

The depositFee is divided by 2 when and sent to 2 addresses. This can result in a value of 1 remaining in the contract, if the depositFee is an odd number.

```
if (_amount > 0) {
    pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
    if (pool.depositFeeBP > 0) {
        uint256 depositFee = _amount.mul(pool.depositFeeBP).div(10000);
        uint256 depositeFeeHalf = depositFee.div(2);
        pool.lpToken.safeTransfer(feeAddBb, depositeFeeHalf);
        pool.lpToken.safeTransfer(feeAddSt, depositeFeeHalf);
        user.amount = user.amount.add(_amount).sub(depositFee);
    } else {
        user.amount = user.amount.add(_amount);
    }
}
```

Example:
User deposits 10130
10130*400/10000 = 405

As solidity does not handle floating points, the result of the deposit fee deposited by 2 will be
405/2 = 202

feeAddBb and FeeAddSt will each receive 202, leaving a balance of 1 in the contract that is not accounted for. This does not affect any user deposits or withdrawals, or affect the user experience.

Recommendations:
Subtract depositeFeeHalf from depositFee to obtain the amount to be sent to the second fee address (feeAddSt).

BLZ-003: Deposit fee can be 0 even if depositFeeBP is non 0

Severity: Infomational

If the deposited amount is smaller than 50, even if the depositFeeBP is non 0, there will be no fees deducted from the user's deposit amount.

Recommendations:
Based on the price of LPs and tokens for the various pools, the gas fee will be more expensive than the amount saved by depositing such a small amount over multiple transactions to avoid the deposit fee. As such, no change needs to be made.

BLZ-004: Inaccurate emission information in Medium post

Severity: Infomational

In the medium post, the emission and distribution information is stated to be the following:

> **Emission Rate:** Reward per block — 1 Blizzard
>
> **Daily Emissions:** 28.8k Blizzard
>
> **Distribution:** Farmers get 0.9112 Blizzard rewards per block, team Yeti gets 0.0888 reward per block.

However, in updatePool(), the code shows the following:

```
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 blzdReward = multiplier.mul(blzdPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
blzd.mint(devaddr, blzdReward.mul(888).div(10000));
blzd.mint(address(this), blzdReward);
pool.accBlzdPerShare = pool.accBlzdPerShare.add(blzdReward.mul(1e12).div(lpSupply));
pool.lastRewardBlock = block.number;
```

The minted amount of blzdPerBlock is 108.8% instead of 100%, which is greater than the information stated in the medium post.

Recommendations:
Update the information in the medium post to reflect the actual calculations in the code.

Note: The team has updated the Medium post to reflect this.

Smart Contract Risk Factors

As the deposit fee for a pool can be increased up to 4%, it is recommended that users verify the deposit fee of the pool they would like to enter before depositing into it.

The owner of the pool can change the pool weightages, but such a transaction to change it has a 24 hour delay. Users can monitor the contract for any queued transactions to add a new pool, or set the pointAlloc for an existing pool.

RECOMMENDATIONS

As it is mentioned in the Medium post that buybacks and funding of special pools will be done with the deposit fees, it is recommended to communicate the use of those funds in a transparent manner.

Overall, we find no issues with the Blizzard.Money smart contracts that are inherent risks with deposits, or detract from the user experience. The issues that are present can be rectified with relatively simple code updates, or changes to the Medium articles.