BATON Security Audit

By csanuragjain

26th May 2023

Summary:

Audit for Baton was conducted between 21st May 2023 - 26th May 2023. Total of 10 vulnerabilities were identified which constituted 3 High, 3 Medium, 1 Low, 1 Gas optimization, 2 Informational

Mitigation review was performed on 31st May 2023. One issue was marked accepted risk and rest all issues have been mitigated.

In Scope Contracts:

https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFactory.sol

Git Commit:

6f01e770e99635de95da79c471a5a269b1b61d3c

Note:

1. Since Baton Monitor functioning was not in scope so proposal creation and their impact on fee changes was not tested
2. A max cap of 25% on fees is expected as confirmed by product team

Disclaimer:

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the project analyzed

Issue List:

H1: LP Withdrawal fees not deducted in withdrawAndNftRemove function
H2: Owner can steal unclaimed rewards
H3: Rewards can stuck if reward rate is not set properly
M1: Reward tokens can stuck permanently
M2: Owner can change reward duration in between Active farming
M3: Staker can lose funds due to Migration
L1: Malicious Farms can steal User funds
G1: Variables can be declared immutable
INFO1: Withdraw Airdropped tokens
INFO2: onlyWhenPoolOver modifier could be revised

H1: LP withdrawal fees not deducted in withdrawAndNftRemove function

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L277-L306

Impact:
batonMonitor will lose the LP Fee on the stake amount

Description:
It was observed that unlike withdraw function, the withdrawAndNftRemove function is missing to deduct batonLPFee while withdrawing

```
function withdrawAndNftRemove(
    uint256 amount,
    uint256 minBaseTokenOutputAmount,
    uint256 minFractionalTokenOutputAmount,
    uint256 deadline
)
    external
    updateReward(msg.sender)
{
    require(amount > 0, "Cannot withdraw 0");
    require(amount <= _balances[msg.sender], "Cannot withdraw more then you have staked");

    // remove the amount the user is unstaking
    _totalSupply = _totalSupply - amount;
    _balances[msg.sender] = _balances[msg.sender] - amount;

    // calculate the amount of base (eth for example) tokens and fractional nft tokens the user should receive
    (uint256 baseTokenOutputAmount, uint256 fractionalTokenOutputAmount) =
        pair.remove(amount, minBaseTokenOutputAmount, minFractionalTokenOutputAmount, deadline);

    // transfer the base / fractional nft tokens to the user
    SafeTransferLib.safeTransferETH(msg.sender, baseTokenOutputAmount);
    ERC20(address(pair)).safeTransfer(msg.sender, fractionalTokenOutputAmount);

    // send user his rewards
    harvest();

    // emit an event
    emit Withdrawn(msg.sender, amount);
}
```

Recommendation:
It is recommended to introduce LP fee on the withdrawn amount in withdrawAndNftRemove function as well

Mitigation Review:
Commit 9f0182fd61813fc4ffea2b3b43f63cfdc426c421 fixes this by introducing the LP fees while withdrawing

```
// if the fee is more then 0 send the fee to batonMonitor
if (batonFeeAmount > 0) {
    stakingToken.safeTransfer(batonMonitor, batonFeeAmount);
}

uint256 amountToRemove = amount - batonFeeAmount;

if (amountToRemove > 0) {
    // calculate the amount of base (eth for example) tokens and fractional nft tokens the user should receive
    (uint256 baseTokenOutputAmount, uint256 fractionalTokenOutputAmount) =
        pair.remove(amountToRemove, minBaseTokenOutputAmount, minFractionalTokenOutputAmount, deadline);

    // transfer the base / fractional nft tokens to the user
    SafeTransferLib.safeTransferETH(msg.sender, baseTokenOutputAmount);
    ERC20(address(pair)).safeTransfer(msg.sender, fractionalTokenOutputAmount);
}

// send user his rewards
harvest();
```

H2: Owner can steal unclaimed rewards

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol

Impact:
Owner can steal User unclaimed rewards

Description:
Seems like farm owner can steal unclaimed rewards by calling notifyRewardAmount with an amount and never pass that amount to the contract. Contract will use available reward balance (which is the unclaimed rewards of users + unearned rewards), giving loss to users

{code}

```
function testPOC1() public {
        vm.startPrank(owner);
        address batonFarmAddress =
            batonFactory.createFarmFromExistingPairETH{ value: 1 ether }(owner,
address(ethPair), weekDuration);
        BatonFarm farm = BatonFarm(payable(batonFarmAddress));

        ethPairLpToken.transfer(user1, 1 ether);
        assertEq(ethPairLpToken.balanceOf(user1), 1 ether);
        vm.stopPrank();

        vm.startPrank(user1);
        ethPairLpToken.approve(address(farm), 1 ether);
        farm.stake(1 ether);

        skip(7 days);
  uint256 initialEarnedBal = farm.earned(user1);
  console.log("User earned %i",initialEarnedBal);
  vm.stopPrank();

  vm.startPrank(owner);
  farm.notifyRewardAmount(1 ether);
  vm.stopPrank();
```

```
  vm.startPrank(user1);
  skip(7 days);
  uint256 initialEarnedBal2 = farm.earned(user1);
  console.log("User earned %i",initialEarnedBal2);
  farm.harvest();
       vm.stopPrank();
    }
```

//This will fail on harvest which means User will lose his rewards

{code}

Recommendation:
Ensure that Rewards are actually transferred to farming contract on calling notifyRewardAmount function

Mitigation Review:
Fixed in commit 24c94acce05bda585744c8bd88f0f663525ab2c6. notifyRewardAmount now transfer funds to itself from msg.sender

H3: Rewards can stuck if reward rate is not set properly

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L431

Impact:
Rewards will stuck permanently in contract

Description:
rewardRate is calculated as reward/duration. Now if duration is X and reward is 2X-1 then rewardRate will be 1. This means almost 50% of rewards will never be distributed. This could happen mostly with tokens with lower decimals like USDC

1. Let's say for 30 days rewards:
   Duration: 2,592,000
   Reward: 2*2,592,000 -1 = 5183999
   Reward Rate: 5183999/2,592,000 = 1

2. Total reward which will be distributed: 2,592,000*1 = 2,592,000

3. This means 2,592,000 -1 reward will stuck in contract

Recommendation:
Owner should be careful while deciding the rewardRate to avoid such conditions. Surplus fund should be sent back to depositor

Mitigation Review:
Changes are made in commit 022325892d1fb7d3055526655dda0a222e9308c8, d440a031e9d7e1beac38a5d133b04de5e5f629dc and 22d56c0a263557251f604d071f479dd8e2e5c1e0.Surplus is calculated for new and existing farming. Any excess surplus is sent back

M1: Reward tokens can stuck permanently

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L388

Impact:
Rewards can be lost

Description:
Reward tokens can stuck permanently if Owner calls migration to same farm contract address

{code}
```
function testMigrateToSelf() public {
        vm.startPrank(owner);
        testFarm.initiateMigration(address(testFarm));
        vm.stopPrank();

        vm.startPrank(batonMonitor);
        testFarm.migrate();
        vm.stopPrank();

  vm.startPrank(owner);
  vm.expectRevert("This contract has been migrated, you cannot deposit new funds.");
        testFarm.initiateMigration(babe);
  vm.expectRevert("This contract has been migrated, you cannot deposit new funds.");
        testFarm.stake(1 ether);

  assertEq(testFarm.rewardsToken().balanceOf(address(testFarm)),100 * 1e6);
    }
```
{code}

Recommendation:
Don't allow self address while migration

Mitigation Review:
Commit 07e0a7a0ae8d23dd4320fb1793eeb6a8c39a8bb4 resolves this by adding a check that migrated contract is not self

{code}
require(_migration != address(this), "Cannot migrate to self");
{code}

M2: Owner can change reward duration in between Active farming

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L435

Impact:
Changing reward duration will impact the amount of reward released per second which will be unexpected by stake users. Stake users will lose rewards since same reward will now be distributed over long timeframe

Description:
If owner calls notifyRewardAmount function with reward 0 for an ongoing farming, then the reward rate will reduce and reward duration will increase which is unexpected

Instead of checking _rewardAmount > 0 in factory (it is not required otherwise also since reward rate check on farm will prevent issue), we can put this check on notifyRewardAmount

{code}
```
function testPOC2() public {
vm.startPrank(owner);
address batonFarmAddress =
batonFactory.createFarmFromExistingPairETH{ value: 1 ether }(owner,
address(ethPair), weekDuration);
BatonFarm farm = BatonFarm(payable(batonFarmAddress));

ethPairLpToken.transfer(user1, 1 ether);
assertEq(ethPairLpToken.balanceOf(user1), 1 ether);
vm.stopPrank();

vm.startPrank(user1);
ethPairLpToken.approve(address(farm), 1 ether);
farm.stake(1 ether);

skip(3 days);
uint256 initialEarnedBal = farm.earned(user1);
console.log("On Day 3, User earned %i",initialEarnedBal);
```

```
vm.stopPrank();

vm.startPrank(owner);
farm.notifyRewardAmount(0 ether);
vm.stopPrank();

vm.startPrank(user1);
skip(4 days);
console.log("On Day 7, User earned %i",farm.earned(user1));
skip(3 days);
console.log("On Day 10, User earned %i",farm.earned(user1));
vm.stopPrank();
}

// Owner increasing a 7 day reward to get distributed in 10 days
{code}
```

Recommendation:
New rewards should be distributed within the remaining farming duration. If farming duration is over then rewards should be distributed over full reward duration

Mitigation Review:
Fixed in Commit 93708a278150715c24b9ce757a0c979b9e0faca4. An existing farm will not extend and use new rewards within leftover duration. If farming period is over then will be treated as new reward

M3: Staker can lose funds due to Migration

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L385

Impact:
If Owner migrates within an Active farm then late stakers will be at loss. This is because on withdraw LP fees will be taken even though late staker have not received any reward due to migration

Description:

1. Staker A stakes amount 100 at block B1
2. In same Block B1, owner decides to migrate the farm
3. All rewards are seized and sent to the new farm
4. Staker A withdraws but have to give LP fees, hence taking loss without earning any reward

Remediation:
Do not deduct LP token fees in case of migration

Mitigation Review:
Accepted Risk

L1: Malicious Farms can steal User funds

Location:
Baton Farm UI

Impact:
Users can invest in malicious farms and lose there staked funds

Description:
The farm creation call is permissionless. Attackers can create a farm with a malicious reward token and pool address.
UI will be showing such malicious farms and users might invest in it since it will be very difficult for users to identify genuine farms

Recommendation:
UI should only show whitelisted farms

Mitigation:
Product team showed that a whitelist json is now maintained. Only farm address mentioned in this whitelist json will be shown on UI which fixes this issue

G1: Variables can be declared immutable

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol

Impact:
Gas saving

Description:
Variables like pair, rewardsToken, stakingToken, batonMonitor, batonFactory should only be initialized within the constructor.
This variables can be defined Immutable in order to prevent some gas

Recommendation:
Declare Variables like pair, rewardsToken, stakingToken, batonMonitor, batonFactory as Immutable

Mitigation Review:
The commit a4346d990bd80f05e3e8dc9b49a488ec3c6d9cda, 76ee4923d83b4e41354170645cda4d16d027b4d1 & 038087dc19c9395b0fa3155d2e1da540511b4057 fixes the issue

INFO1: Withdraw Airdropped tokens

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol

Impact:
If Airdrop is made on the farming contract then there is no way to extract the same

Description:
If there is an airdrop on the farming contract then there is no way to extract the airdropped tokens

Recommendation:
An additional Owner withdraw function to withdraw any airdropped token which is not staked or reward token

Mitigation Review:
Fixed in commit 0bab8351dfcc211614b3515798b6366efe7ad1ff, a5e455770af669903443b6594f7064dac45eeced, 10c9eca51c6b57f5c6ddd397da5859f0079348dc. Token which is not staking and reward can be recovered by transferring to owner

INFO2: onlyWhenPoolOver modifier could be revised

Location:
https://github.com/baton-finance/baton-contracts/blob/6f01e770e99635de95da79c471a5a269b1b61d3c/src/BatonFarm.sol#L555-L558

Impact:
User will get no reward if he stakes at periodFinish. This modifier could be revised to show the same

Description:

Although this modifier is not used anywhere but from comment this seems to signify timeline when the farm is active.
This is not true when block.timestamp = periodFinish since User will not get any staking reward from this time

```
/**
 * @notice Modifier to ensure that the pool is still active before calling a function.
 * @dev Requires that the migration has not been completed.
 */
modifier onlyWhenPoolOver() {
    require(block.timestamp > periodFinish, "This farm is still active");
    _;
}
```

Recommendation:
Change onlyWhenPoolOver from

{code}
– – require(block.timestamp > periodFinish, "This farm is still active");
++ require(block.timestamp >= periodFinish, "This farm is still active");
{code}

Mitigation Review:
The issue is now resolved in commit 29093170e685682e2679035883275d6dba87ce82. Check has now been revised to

{code}
require(block.timestamp >= periodFinish, "This farm is still active")

{code}