Smart Contract Audit Report

Provided for:  Baton

Auditor: Jessica Pointing

18 May 2023

1. Executive Summary

1.1. Project

Baton is a yield farming protocol for NFT AMMs. The audit was conducted Saturday 13 May - Thursday 18 May 2023.

1.2. Issues

Total Issues	40
Critical	4
High	7
Medium	6
Low	6
Informational	9
Spelling	8

1.3. Methods

Manual Review
The auditor manually reviewed every line of code in the contracts in scope.

Architecture Overview
The auditor created an architecture overview of the system, including an overview of the functions, function visibilities, variables, variable visibilities, assets, states, and roles of the system.

Unit Testing
The auditor created a few unit tests to verify functionality of functions and provide proof-of-concept for issues found.

Automated tool testing
The auditor used Slither, which is an automated tool that runs a suite of vulnerability detectors and prints visual information about contract details.

Formal Verification
The auditor used formal verification, which can make proofs that certain properties of the system holds. The auditor used the Certora Prover Language to write formal verification rules.

1.4. Scope

This audit covered the following files:

File	Lines of Code	Description
BatonFarm.sol	474	A yield farming platform that allows users to stake NFT AMM LP positions and earn rewards
BatonFactory.sol	183	A factory for creating BatonFarms with different reward types (ERC20, ETH, fractional NFTs)

2. System Overview

2.1. Roles

- Farm Owner
- BatonMonitor
- rewardsDistributor

2.2. States of the system

Variable	Value
PeriodFinish	0
PeriodFinish	> block.timestamp
PeriodFinish	< block.timestamp
PeriodFinish	= block.timestamp
MigrationComplete	FALSE
MigrationComplete	TRUE
FeeProposalApprovalDate	0

FeeProposalApprovalDate	> block.timestamp
FeeProposalApprovalDate	< block.timestamp
FeeProposalApprovalDate	= block.timestamp
Paused	FALSE
Paused	TRUE

2.3. Actionable Functions

Staking functions

Function	Role
Stake	Msg.sender
NftAddAndStake	Msg.sender
WithdrawAndNftRemove	Msg.sender
Withdraw	Msg.sender
Harvest	Msg.sender

Migration functions

Function	Role
InitiateMigration	Owner
Migrate	BatonMonitor

Fee functions

Function	Role
ProposeNewFee	BatonMonitor
SetFeeRate	BatonMonitor

Reward functions

Function	Role
NotifyRewardAmount	RewardsDistributor, Owner
SetRewardsDistributor	Owner

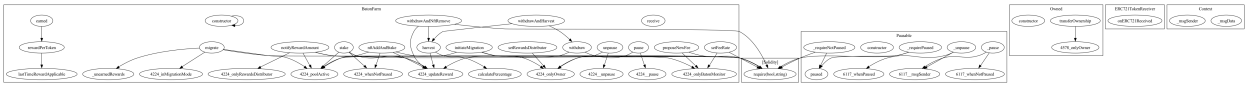
Pause functions

Function	Role
Pause	Owner
Unpause	Owner

2.4. Assets

- rewardsToken
- StakingToken
- Caviar Pair
- ETH

2.5. Graph of BatonFarm contract



3. Summary of Findings

Critical Risk Findings

ID	Description	File(s)	LOC	Severity
4.1	Rewards could be locked in contract and therefore lost	BatonFarm.sol	167,171	**Critical**
4.2	NFTs are transferred from owner instead of msg.sender	BatonFactory.sol	158	**Critical**
4.3	Creating farms with NFTs does not separate the underlying staking pool and reward asset	BatonFactory.sol	179	**Critical**

4.4	Reentrancy in migrate	BatonFarm.sol	366	**Critical**

High Risk Findings

ID	Description	File(s)	LOC	Severity
5.1	Lack of zero address check for rewardsDistributor, owner, batonMonitor	BatonFarm.sol	80	**High**
5.2	Lack of check that rewardsDistributor could be set to zero	BatonFarm.sol	433	**High**
5.3	User can not withdraw their NFTs	BatonFarm.sol	272	**High**
5.4	Lack of check that baseTokenOutputAmount and fractionalTokenOutputAmount received	BatonFarm.sol	273	**High**
5.5	Lack of check that pairAddress is valid Caviar pair	BatonFactory.sol	76,110	**High**
5.6	Additional rewards that are added could be lost	BatonFarm.sol	-	**High**
5.7	Reentrancy in NftAddAndStake	BatonFarm.sol	230	**High**

Medium Risk Findings

ID	Description	File(s)	LOC	Severity
6.1	Creating a farm could result in denial-of-service due to revert or block gas limit when looping over NFTs	BatonFactory.sol	157-159	**Medium**
6.2	Staking NFTs could result in denial-of-service due to revert or block gas limit when looping over NFTs	BatonFarm.sol	225-227	**Medium**
6.3	No Baton Fee for first seven days of farm	BatonFarm.sol	390	**Medium**
6.4	User can still stake after farm has been shutdown	BatonFarm.sol	190	**Medium**
6.5	No check that rewardsDuration is equal to zero that could lead to denial of service	BatonFarm.sol	418, 422	**Medium**
6.6	Function missing to set rewards duration	BatonFarm.sol	-	**Medium**

Low Risk Findings

ID	Description	File(s)	LOC	Severity
7.1	No check that reward to be sent to user is not equal to zero	BatonFarm.sol	324	**Low**
7.2	No event emitted when contract receives ETH	BatonFarm.sol	107	**Low**
7.3	Contract address is passed as argument to function	BatonFarm.sol	55	**Low**
7.4	No check that reward is equal to zero	BatonFactory.sol	75, 118, 147	**Low**
7.5	No event emitted for setRewardsDistributor	BatonFarm.sol	443	**Low**
7.6	No minimum and maximum rewardsDuration	BatonFarm.sol	96	**Low**

Informational Findings

ID	Description	File(s)	LOC	Type
8.1	Function calculatePercentage could be pure	BatonFarm.sol	119	**Informational**
8.2	Multiple functions could be declared as external	BatonFarm.sol, BatonFactory.sol	-	**Informational**
8.3	Variable balance is not used	BatonFarm.sol	429	**Informational**
8.4	Variable balanceOfAccount is not necessary	BatonFarm.sol	157	**Informational**
8.5	Event RewardsDuration is not used	BatonFarm.sol	71	**Informational**
8.6	Event Recovered is not used	BatonFarm.sol	72	**Informational**
8.7	Remove public for constructors in BatonFarm and BatonFactory	BatonFarm.sol, BatonFactory.sol	80, 55	**Informational**
8.8	RewardPerToken is shadowed declaration	BatonFarm.sol	155, 136-147	**Informational**
8.9	State variables could be private	BatonFarm.sol	-	**Informational**

Spelling Findings

ID	Mistake	Correction	File(s)	LOC	Type
9.1	CreateFarmFromExsistingPairNFT	CreateFarmFromExistingPairNFT	BatonFactory.sol	144	**Spelling**
9.2	CreateFarmFromExsistingPairERC20	CreateFarmFromExistingPairERC20	BatonFactory.sol	72	**Spelling**
9.3	Fun	Fund	BatonFactory.sol	86	**Spelling**
9.4	Propse	Propose	BatonFarm.sol	52	**Spelling**
9.5	Amout	Amount	BatonFarm.sol	267, 271	**Spelling**
9.6	Recive	Receive	BatonFarm.sol	271	**Spelling**
9.7	An	A	BatonFarm.sol	411	**Spelling**
9.8	Contract	Address	BatonFarm.sol	358	**Spelling**

Proven Properties of System

ID	Description	Type
10.1	Staking increases total supply	**Proven**
10.2	Withdrawing decreases total supply	**Proven**

4. **Critical** Risk Findings

4.1. Rewards could be locked in contract and therefore lost

Severity	File(s) affected	Line(s) of code
Critical	BatonFarm.sol	167, 171

Description

When the BatonMonitor calls the migrate function in order to migrate the rewards in the farm contract to a new migration address, the rewardsToMigrate are calculated by the

_unearnedRewards. The _unearnedRewards function, however, returns zero if the totalSupply is equal to zero and if the block.timestamp >= periodFinish.

Impact

This means that even if there are rewardTokens in the contract, it would not be possible to withdraw the rewards from the contract. Once the migration is complete, it is not possible to update the periodFinish. The funds would be locked in the contract and therefore lost.

Proof of Concept

Below is a test case that correctly migrates the rewardsTokens to the migration address when totalSupply != 0 and when block.timestamp <= periodFinish.

```
function testShouldMigrateRewards() public {
        vm.startPrank(owner);
        testFarm.initiateMigration(babe);
        ercPairLpToken.approve(address(testFarm), 100 ether);
        vm.stopPrank();

        vm.startPrank(user1);
        ercPairLpToken.approve(address(testFarm), 10 ether);
        testFarm.stake(1 ether);
        vm.stopPrank();

        vm.startPrank(batonMonitor);
        uint256 testFarmRewardsBalanceBefore =
        testFarm.rewardsToken().balanceOf(address(testFarm));
        console.log(testFarmRewardsBalanceBefore);
        uint256 totalSupplyBefore = testFarm.totalSupply();
        console.log(totalSupplyBefore);
        uint256 migrationRewardsBalanceBefore =
        testFarm.rewardsToken().balanceOf(babe);
        uint256 rewardsToMigrate = testFarm._unearnedRewards();
        console.log(rewardsToMigrate);
        testFarm.migrate();
        uint256 migrationRewardsBalanceAfter =
        testFarm.rewardsToken().balanceOf(babe);
        uint256 testFarmRewardsBalanceAfter =
        testFarm.rewardsToken().balanceOf(address(testFarm));
        vm.stopPrank();

        console.log(testFarmRewardsBalanceBefore);
        console.log(testFarmRewardsBalanceAfter);

        console.log(migrationRewardsBalanceBefore);
        console.log(migrationRewardsBalanceAfter);

        assertGt(migrationRewardsBalanceAfter, migrationRewardsBalanceBefore);
        assertLt(testFarmRewardsBalanceAfter, testFarmRewardsBalanceBefore);

        vm.startPrank(address(batonFactory));
        vm.expectRevert("This contract has been migrated, you cannot deposit new
        funds.");
        testFarm.notifyRewardAmount(1 ether);
}
```

Now I modify the test case so that the totalSupply = 0. The test fails because the rewards can not be migrated.

```
  function testShouldNotLockRewardTokens() public {

      vm.startPrank(owner);
      testFarm.initiateMigration(babe);
```

```
        ercPairLpToken.approve(address(testFarm), 100 ether);
        vm.stopPrank();

        vm.startPrank(user1);
        ercPairLpToken.approve(address(testFarm), 10 ether);
        // testFarm.stake(1 ether);
        vm.stopPrank();

        vm.startPrank(batonMonitor);
        uint256 testFarmRewardsBalanceBefore = testFarm.rewardsToken().balanceOf(address(testFarm));
        console.log(testFarmRewardsBalanceBefore);
        uint256 totalSupplyBefore = testFarm.totalSupply();
        console.log(totalSupplyBefore);
        uint256 migrationRewardsBalanceBefore = testFarm.rewardsToken().balanceOf(babe);
        uint256 rewardsToMigrate = testFarm._unearnedRewards();
        console.log(rewardsToMigrate);
        testFarm.migrate();
        uint256 migrationRewardsBalanceAfter = testFarm.rewardsToken().balanceOf(babe);
        uint256 testFarmRewardsBalanceAfter = testFarm.rewardsToken().balanceOf(address(testFarm));
        vm.stopPrank();

        console.log(testFarmRewardsBalanceBefore);
        console.log(testFarmRewardsBalanceAfter);

        console.log(migrationRewardsBalanceBefore);
        console.log(migrationRewardsBalanceAfter);

        assertGt(migrationRewardsBalanceAfter, migrationRewardsBalanceBefore);
        assertLt(testFarmRewardsBalanceAfter, testFarmRewardsBalanceBefore);

        vm.startPrank(address(batonFactory));
        vm.expectRevert("This contract has been migrated, you cannot deposit new funds.");
        testFarm.notifyRewardAmount(1 ether);
    }
```

Recommendation

Implement a recoverFunds function instead of the migrate and _unearnedRewards functions as implemented in the Synthetix contract: https://github.com/Synthetixio/synthetix/blob/develop/contracts/StakingRewards.sol

```
// Added to support recovering LP Rewards from other systems such as BAL to be distributed to holders

function recoverERC20(address tokenAddress, uint256 tokenAmount) external onlyOwner {
        require(tokenAddress != address(stakingToken), "Cannot withdraw the staking token");
        IERC20(tokenAddress).safeTransfer(owner, tokenAmount);
        emit Recovered(tokenAddress, tokenAmount);
}
```

4.2. NFTs are transferred from owner instead of msg.sender

Severity	File(s) affected	Line(s) of code
Critical	BatonFactory.sol	158

Description

NFTs are transferred from owner instead of BatonFactory

Impact

This would transfer the NFTs from the owner unknowingly, causing a loss of their assets

Recommendation

Transfer NFTs from msg.sender

4.3. Creating farm with NFTs does not separate the underlying staking pool and the reward asset

Severity	File(s) affected	Line(s) of code
Critical	BatonFactory.sol	179

Description

In the function `createFarmFromExistingPairNFT`, the farm is created with the same underlying staking pool and the same rewards NFT asset.

Impact

The user would create a farm for the wrong underlying staking pool.

Recommendation

Separate the receiving pool and reward asset pair.

4.4. Reentrancy in migrate

Severity	File(s) affected	Line(s) of code
Critical	BatonFarm.sol	366

Description

MigrationComplete and periodFinish variables are updated after the transferring of tokens

Impact

The user who calls migrate can perform a reentrancy attack, draining all of the rewardsToken funds, including those already earned by users

Recommendation

Update state variables before

5. High Risk Findings

5.1. Lacks zero address check for rewardsDistributor, batonMonitor, owner

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	80

Description

When the BatonFarm is created, there are no checks to ensure that the owner, the rewardsDistributor, and batonMonitor are not the zero address.

Impact

If any of these roles are the zero address, then the farm has lost some of its functionality and it would not be possible to recover any funds in the contract.

Recommendation

Add zero address checks: e.g. `batonMonitor != address(0)`

5.2. Lack of check that rewardsDistributor could be set to zero

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	443

Description

The function setRewardsDistributor sets the rewardsDistributor but does not check that the rewardsDistributor is set to the zero address

Impact

The rewardsDistributor is responsible for notifying the reward amount.

Recommendation

Add a zero address check

5.3. User can not withdraw their NFTs

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	272

Description

The withdrawAndNFTRemove only returns the baseTokenOutputAmount and fractionalTokenOutputAmount but not the NFTs they transferred

Impact

The user would not be able to withdraw their NFTs

Recommendation

Add a withdraw function to withdraw NFTs

5.4. Lack of check that baseTokenOutputAmount and fractionalTokenOutputAmount received

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	273

Description

The amounts of baseTokenOutputAmount and fractionalTokenOutputAmount are sent to the msg.sender but there is no check that the farm contract actually received those correct amount.

Impact

The transfer could revert of the baseTokenOutputAmount and fractionalTokenOutputAmount funds of the farm contract could be drained.

Recommendation

Add a check that the contract receives the tokens.

5.5. Lack of check pair address is valid Caviar Pair

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	76, 110

Description

The user could input a pair address that is not a valid Caviar pair

Impact

The user could set up an invalid pair address and manipulate some of the functions

Recommendation

Add a check that the pair address comes from Caviar

5.6. Additional rewards that are added could be lost

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	-

Description

Someone could send rewardsTokens to the contract but they could be locked in the contract as they could not withdraw them

Impact

rewardsTokens are locked in the contract

Recommendation

After the farm has shutdown, remaining rewardsTokens (minus those earned) could be claimed by owner

5.7. Reentrancy in NftAddAndStake

Severity	File(s) affected	Line(s) of code
High	BatonFarm.sol	230

Description

An external call to retrieve the lpTokenAmount is called before the totalSupply and balanceOfUser is updated. A malicious contract could keep calling nftAddAndStake, but the totalSupply would be lower than it should be, leading to a higher rewards amount.

Impact

Malicious actor could claim higher rewards

Recommendation

Retrieve the lpTokenAmount making the transfer and update the state variables totalSupply and balance before the transfer.

6. Medium Risk Findings

6.1. Creating a farm could result in denial-of-service due to revert or block gas limit when looping over NFTs

Severity	File(s) affected	Line(s) of code
Medium	BatonFactory.sol	157-159

Description

The function `createFarmFromExsistingPairNFT` loops over NFTs and transfers them from the owner of the farm to the BatonFactory contract. If one of the transfers fails, then the function would revert and the functionality of the function is lost. In addition, if the number of NFTs to transfer is large, then the gas block limit will be reached and the functionality would be lost.

Impact

The user would not be able to create a farm with their NFTs.

Proof of Concept

Below is a test case that shows that with 500 NFTs, the gas block limit of 30M would be reached. On mainnet, this would result in a failed transaction.

```
    function testCreateFarmFromExsistingPairNFTLarge() public {
        // (gas: 33098504)
        vm.startPrank(owner);
        bayc.setApprovalForAll(address(batonFactory), true);
        for (uint256 i = 5; i < (5*1e2+5); i++) {
            bayc.mint(owner, i);
            tokenIdsToStake.push(i);
        }

        address batonFarmAddress = batonFactory.createFarmFromExsistingPairNFT(
            owner, address(bayc), tokenIdsToStake, weekDuration, reservoirOracleMessages
        );
        BatonFarm farm = BatonFarm(payable(batonFarmAddress));

        vm.stopPrank();

        assertEq(ERC20(address(farm.rewardsToken())).balanceOf(address(farm)), 5*1e2 ether);
        assertEq(farm.lastUpdateTime(), block.timestamp);
        assertEq(farm.periodFinish(), block.timestamp + farm.rewardsDuration());
        assertEq(farm.rewardRate(), 5*1e2 ether / weekDuration);
    }
```

Recommendation

Loop over the NFTs in batches, in which each batch is safely below the gas block limit and keep track of the position in the iteration to resume from that point. See template of an example below.

```
struct Payee {
    address addr;
    uint256 value;
}

Payee[] payees;
uint256 nextPayeeIndex;

function payOut() {
    uint256 i = nextPayeeIndex;
    while (i < payees.length && msg.gas > 200000) {
      payees[i].addr.send(payees[i].value);
      i++;
    }
    nextPayeeIndex = i;
}
```

6.2. Staking NFT could result in denial-of-service due to revert of block gas limit when looping over NFTs

Severity	File(s) affected	Line(s) of code
Medium	BatonFarm.sol	225-227

Description

This issue is similar to issue 6.1. The function `nftAddAndStake` loops over NFTs and transfers them from the message sender to the BatonFarm contract. If one of the transfers fails, then the function would revert and the functionality of the function is lost. In addition, if the number of NFTs to transfer is large, then the gas block limit will be reached and the functionality would be lost.

Impact

The user would not be able to stake their NFTs.

Recommendation

Loop over the NFTs in batches, in which each batch is safely below the gas block limit and keep track of the position in the iteration to resume from that point. See code example in 6.1

Note: This issue is also present in the Caviar contract.

6.3. No Baton Fee for first seven days of farm

Severity	File(s) affected	Line(s) of code
Medium	BatonFarm.sol	390

Description

The BatonFee is initialised to zero but the only way to change the BatonFee is to call the proposeNewFee function and then the setFeeRate function but seven days have to pass for the fee rate to be set.

Impact

The BatonMonitor would lose the fees accumulated during the first seven days of the farm.

Recommendation

Initialise the BatonFee to an amount and not use the default initialisation of zero or set the BatonFee in the BatonFactory.

6.4. User can still stake after the farm has been shutdown

Severity	File(s) affected	Line(s) of code
Medium	BatonFarm.sol	190, 209

Description

The farm has been shutdown when periodFinish is in the past. The stake function does not check whether the farm has been shutdown and therefore a user can still stake.

Impact

The user would be able to stake but would not receive any rewards as the farm has been shutdown.

Recommendation

Check that the farm has not been shutdown in the staking functions.

6.5. No check that rewardsDuration is not equal to zero, which could lead to denial-of-service

Severity	File(s) affected	Line(s) of code
Medium	BatonFarm.sol	418, 422

Description

rewardsDuration is set in the constructor but there is no check to require that the rewardsDuration is greater than zero. If rewardsDuration is equal to zero, then there will be a division by zero error in lines 418 and 422 in the notifyRewardAmount function.

Impact

The farm would not be functional

Recommendation

Add a require statement that rewardsDuration is greater than zero.

6.6. Function missing to set rewards duration

Severity	File(s) affected	Line(s) of code
Medium	BatonFarm.sol	-

Description

There is no ability to set the rewards duration after farm has been created

Impact

The farm rewards duration can not be changed

Recommendation

Add setRewardsDuration function. See example below from Synthetix: https://github.com/Synthetixio/synthetix/blob/develop/contracts/StakingRewards.sol

```
function setRewardsDuration(uint256 _rewardsDuration) external onlyOwner {
    require(
        block.timestamp > periodFinish,
        "Previous rewards period must be complete before changing the duration for the new period"
    );
    rewardsDuration = _rewardsDuration;
    emit RewardsDurationUpdated(rewardsDuration);
}
```

7. Low Risk Findings

7.1. No check that reward to be sent to user is not equal to zero

Severity	File(s) affected	Line(s) of code
Low	BatonFarm.sol	324

Description

In the harvest function, there is a check that the batonFeeAmount is greater than zero before transferring it to the batonMonitor. There is no check, however, that the reward to be sent to the user (the batonFeeAmount subtracted from the reward) is greater than zero.

Impact

This means that a zero amount of tokens could be transferred to the user. In addition an event is emitted stating “RewardPaid” even with a zero amount.

Recommendation

Add a require statement that the reward - batonFeeAmount is greater than zero. See code example below.

```
function harvest() public updateReward(msg.sender) {
        uint256 reward = rewards[msg.sender]; // get users earned fees
        uint256 batonFeeAmount = calculatePercentage(batonFee, reward); // calculate batons fee

        rewards[msg.sender] = 0; // clear the reward counter for the user

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

        // send the reward - batonFeeAmount to msg.sender
        // if the fee is 0 it wont effect the amount set to msg.sender

        if ((reward - batonFeeAmount) > 0) {
            rewardsToken.safeTransfer(msg.sender, reward - batonFeeAmount);

        }
        // emit an event
        emit RewardPaid(msg.sender, reward - batonFeeAmount);
    }
```

7.2. No event emitted when contract receives ETH

Severity	File(s) affected	Line(s) of code
Low	BatonFarm.sol	107

Description

In the receive function, no event is emitted.

Recommendation

Emit an event when the contract receives. See code example below.

```
event Received(address, uint);

receive() external payable {
        emit Received(msg.sender, msg.value);
}
```

7.3. Contract address is passed as argument to function

Severity	File(s) affected	Line(s) of code
Low	BatonFarm.sol	55

Description

A contract address is passed as an argument to the function, but it would be better to pass the interface rather than the address for additional type safety guarantees.

Recommendation

Pass interface instead of contract address

7.4. No check that reward is equal to zero

Severity	File(s) affected	Line(s) of code
Low	BatonFactory.sol	75, 188, 147

Description

There is no check that the user creates a farm with zero rewards. Also if there is no minimum reward amount, users could spam the system creating many farms with low rewards.

Recommendation

Add a zero check

7.5. No event emitted for setRewardsDistributor

Severity	File(s) affected	Line(s) of code
Low	BatonFarm.sol	443

Description

There is no event when the rewardsDistributor is set

Recommendation

Add an event

7.6. No minimum and maximum rewardsDuration

Severity	File(s) affected	Line(s) of code
Low	BatonFarm.sol	96

Description

No minimum and maximum rewardsDuration

Recommendation

Add minimum and maximum rewardsDuration

8. Informational Findings

8.1. Function calculatePercentage could be pure

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	119

8.2. Multiple functions could be declared as external

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	-

The following functions could be declared as external:
- stake
- NftAddAndStake
- WithdrawAndNftRemove
- WithdrawAndHarvest
- InitiateMigration
- Migrate
- ProposeNewFee
- SetFeeRate
- NotifyRewardAmount
- SetRewardsDistributor
- Pause
- Unpause
- CreateFarmFromExistingPairERC20
- CreateFarmFromExistingPairETH
- CreateFarmFromExistingPairNFT

8.3. Variable balance is not used

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	429

8.4. BalanceOfAccount variable is not necessary

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	157

8.5. Event RewardsDuration is not used

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	71

8.6. Event Recovered is not used

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	72

8.7. Remove public from constructors in BatonFarm and BatonFactory

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol, BatonFactory.sol	80, 55

8.8. RewardPerToken is shadowed declaration

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	155, 136-147

8.9. State variables could be private

Severity	File(s) affected	Line(s) of code
Informational	BatonFarm.sol	34, 42

totalSupply and balance could be private state variables

9. **Spelling** Findings

ID	Mistake	Correction	File(s)	LOC	Type
9.1	CreateFarmFromExsistingPairNFT	CreateFarmFromExistingPairNFT	BatonFactory.sol	144	**Spelling**
9.2	CreateFarmFromExsistingPairERC20	CreateFarmFromExistingPairERC20	BatonFactory.sol	72	**Spelling**
9.3	Fun	Fund	BatonFactory.sol	86	**Spelling**
9.4	Propse	Propose	BatonFarm.sol	52	**Spelling**
9.5	Amout	Amount	BatonFarm.sol	267, 271	**Spelling**
9.6	Recive	Receive	BatonFarm.sol	271	**Spelling**
9.7	An	A	BatonFarm.sol	411	**Spelling**
9.8	Contract	Address	BatonFarm.sol	358	**Spelling**

10. **Proven** Properties of System

10.1. Staking increases total supply

```
rule stakingIncreasesTotalSupply(method f){
    env e;
    uint256 amount;
    uint256 totalSupplyBefore = totalSupply();
    stake(e, amount);
    uint256 totalSupplyAfter = totalSupply();

    assert totalSupplyBefore <= totalSupplyAfter;
}
```

10.2. Withdraw decreases total supply

```
rule withdrawDecreasesTotalSupply(method f){
    env e;
    uint256 amount;
    uint256 totalSupplyBefore = totalSupply();
    withdraw(e, amount);
    uint256 totalSupplyAfter = totalSupply();

    assert totalSupplyBefore >= totalSupplyAfter;

}
```

11. Disclaimer

This audit report may not include all vulnerabilities of the code and does not guarantee its security. The report is based on the limited scope described in Section 1.4 and was conducted during a limited timeframe with limited materials and documentation. The protocol may choose to change

the code at a later date, which could introduce new vulnerabilities. The protocol relies on other protocol implementations, including but not limited to Caviar, which is out of scope for this audit. Any vulnerabilities in libraries that Baton depends on, including Caviar's protocol, could lead to vulnerabilities in Baton.

Your access and/or use of this protocol will be at your sole risk. This report does not indicate the endorsement of any particular project or team. 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. The report, its content, access, and/or usage thereof, including any associate services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory or other advice. The Auditor shall not be liable in any way in respect of, including but not limited to, any loss of production, profit or use, loss of funds for Baton, the protocol, its users, any other stakeholders, or any indirect or consequential damage or claim arising in connection with this audit report.