VotingEscrowTruf::stake() do not hold the returned lockupId

Summary

When the stake() is called by one user to stake for another user this function do not holds the returned value which is lockupId of the position.

Vulnerability Detail

The internal _stake() returns lockupId for a staked position, this function is called by stake() but this stake() does not hold the lockupId.

Impact

Calling this function will not return lockupId of the position.

Code Snippet

```
function stake(uint256 amount, uint256 duration, address to) external {
    _stake(amount, duration, to, false); // @audit-issue _stake() returns a
lockUp Id, but here is nothing to hold it
  }
```

Tool used

Manual Review

Recommendation

Replace this function with this:

```
function stake(uint256 amount, uint256 duration, address to) external
returns(uint256 lockupId) {
   lockupId =  _stake(amount, duration, to, false);
  }
```

TrufVesting::onlyOwner modifier should be removed for setUserVesting()

Summary

The function setUserVesting() is controlled by owner i.e a single authority which could potentially be a centralization risk.

Vulnerability Detail

An important component of setUserVesting() is amount which is passed by owner. This amount is TRUF token, the token amount is modified in this function. But the issue is an user will have to blindly trust the authority to set this information which brings centralization risk.

Impact

The amount could be not the choice of the staker. Also user may not want to set vesting amount.

Tool used

Manual Review

Recommendation

Remove the onlyOwner modifier for this function and let users handle this operation.

VotingEscrowTurf::incorrect reward distribution

Summary

In VotingEscrowTurf.sol contract rewards is distributed in such a way that only first staker will get maximum reward and subsequent stakers will get 0 reward no matter how long they have been staking.

Severity

High

Vulnerability Detail

Here is the test:

```
function test_rewardDistributionOccuringIncorrectly() external {
    console.log("trufStakingReward's balance: ",
trufToken.balanceOf(address(trufStakingRewards)));
    vm.startPrank(alice);
    veTRUF.stake(1000, 10 days);
    vm.warp(block.timestamp + 10 days);
    console.log('reward before claiming: ',
trufStakingRewards.earned(alice));
    console.log('before claiming the reward token balance of alice is : ',
trufToken.balanceOf(alice));
    veTRUF.claimReward();
    console.log('after claiming the reward token balance of alice is : ',
trufToken.balanceOf(alice));
    veTRUF.unstake(0);
    console.log("Alice unstaked her position");
    vm.stopPrank();
    console.log("trufStakingReward's balance after unstaking of Alice: ",
trufToken.balanceOf(address(trufStakingRewards)));
    vm.startPrank(bob);
    trufToken.approve(address(veTRUF), type(uint256).max);
    console.log("Bob staked");
    veTRUF.stake(1000, 100 days);
    vm.warp(block.timestamp + 100 days);
    console.log("Bob's earned reward is: ",
trufStakingRewards.earned(bob));
    veTRUF.unstake(0);
    console.log("Bob unstaked");
    console.log("trufStakingReward's balance after unstaking of Bob: ",
trufToken.balanceOf(address(trufStakingRewards)));
    vm.startPrank(carol);
    trufToken.approve(address(veTRUF), type(uint256).max);
    console.log("Carol staked");
    veTRUF.stake(1000, 100 days);
    vm.warp(block.timestamp + 100 days);
    console.log("Carol's earned reward is: ",
trufStakingRewards.earned(carol));
    veTRUF.unstake(0);
    console.log("Carol unstaked");
    console.log("trufStakingReward's balance after unstaking of Carol: ",
trufToken.balanceOf(address(trufStakingRewards)));

  }
```

If we run this test we get:

```
[PASS] test_rewardDistributionOccuringIncorrectly() (gas: 996043)
Logs:
  trufStakingReward's balance:  200000000000000000000
  reward before claiming:  199999999999999584000
  before claiming the reward token balance of alice is :
3333333333333333333332333
  after claiming the reward token balance of alice is :
3333335333333333333332916333
  Alice unstaked her position
  trufStakingReward's balance after unstaking of Alice:  416000
  Bob staked
  Bob's earned reward is:  0
  Bob unstaked
  trufStakingReward's balance after unstaking of Bob:  416000
  Carol staked
  Carol's earned reward is:  0
  Carol unstaked
  trufStakingReward's balance after unstaking of Carol:  416000

Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 2.28ms
```

So what is happening here? One thing to note that reward is distributed by VirtualStakingRewards.sol i.e in this test trufStakingRewards contract. So first we can see the balance of trufStakingReward contract is : 20000000000000000000. Then Alice staked 1000 Truf token for 10 days, after 10 days the claimable reward for Alice is: 19999999999999584000. You can see that surprisingly this amount is very high. If we substract this reward amount from the balance of trufStakingRewards contract then we will get only: 416000. So in trufStakingRewards contract ony 416000 Truf token is remaining to give reward, but the way reward is calculating and given away this amount is not sufficient, for that reason subsequent stakers like Bob, Carol are not getting any reward although they are staking for far longer time than Alice. Even if Alice do not claim the reward and do not unstake still she will earn same amount of reward and Bob, Carol will not get any reward. See this in test:

```
function test_rewardDistributionOccuringIncorrectly2() external {
    console.log("trufStakingReward's balance: ",
trufToken.balanceOf(address(trufStakingRewards)));
    vm.prank(alice);

    veTRUF.stake(50e18, 10 days);
    console.log(' Alice staked');
    vm.warp(block.timestamp + 10 days);
    console.log("Alice's earned reward: ",
trufStakingRewards.earned(alice));
    vm.startPrank(bob);
    trufToken.approve(address(veTRUF), type(uint256).max);

    veTRUF.stake(50e18, 100 days);
    console.log('Bob staked');
    vm.warp(block.timestamp + 100 days);
    console.log("Bob's earned reward is: ",
trufStakingRewards.earned(bob));
    vm.stopPrank();
    vm.startPrank(carol);
    trufToken.approve(address(veTRUF), type(uint256).max);

    veTRUF.stake(1000, 100 days);
    console.log('Carol staked');
    vm.warp(block.timestamp + 100 days);
    console.log("Carol's earned reward is: ",
trufStakingRewards.earned(carol));
  }
```

And result is:

```
[PASS] test_rewardDistributionOccuringIncorrectly2() (gas: 987983)
Logs:
  trufStakingReward's balance:  20000000000000000000
   Alice staked
  Alice's earned reward:  19999999999999583999
  Bob staked
```

```
    Bob's earned reward is:  0
    Carol staked
    Carol's earned reward is:  0

  Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.74ms
```

Impact

All stakers after first staker will not get any reward.