Nexus Mutual Pooled Staking / June 2020

Files in scope

```
contracts/PooledStaking.sol
```

The audit was done in three phases:

First phase: https://github.com/NexusMutual/smart-contracts/commit/d0120419b21dbd2bd8469214c0e43fa146bd9cea (issues 1 - 2)

Second phase: https://github.com/NexusMutual/smart-contracts/commit/b2b3e58eeaf8305aebf733c916c5eec9b1d868a4 (issues 3 - 5)

Third phase: https://github.com/NexusMutual/smart-contracts/commit/ce0b34acf71488f5316c274cf203b675d32ad872 (issues 6 - 7)

In addition integration of the contract into this system has been audited at this commit

https://github.com/NexusMutual/smart-contracts/tree/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

Current status

As of June 29th all raised issues have been fixed by the developer

Issues

1. Stake function accepts duplicate records in the `_contracts` array

type: security / severity: critical

In the `stake` function `_contracts` array can contain duplicate addresses, this allows users to create multiple records of their address in the `contracts[contractAddress].stakers` array which will lead to them being able to collect rewards multiple times on the same effective stake.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

2. Segmented processing of burn command updates values that are expected to stay constant during the processing

type: security / severity: critical

Due to `_contract.staked` being updated mid way through processing burn if there's not enough gas, calculation of burned amount can become incorrect. For example `_contract.staked` on `line 532` can become lower than `burn.amount`.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

3. In depositAndStake, stake amounts are not counted in the total stake if they have not been updated

type: security / severity: critical

In `depositAndStake` if the value of new stake is the same as the old stake, the stake is not added in the `totalStaked` which allows users to bypass limit on maximum leverage.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

4. Intermediate value of a variable in segmented processing function is not stored

type: correctness / severity: critical

`_stakedOnContract` value in `_calculateContractStake` is not stored if the cycle doesn't finish.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

5. In depositAndStake, stake amount is not updated when new stake is equal to effective old stake (limited by deposit)

type: correctness / severity: major

In the `depositAndStake` function, if `oldStake == newStake` but `initialStake` is higher, after deposit is increased the effective stake will no longer be equal to the `newStake`. This hypothetically alows users to bypass limit on maximum leverage, but also leads to other issues.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

6. Values in segmented processing are not cleaned after task has been finished, contaminating subsequent tasks

type: correctness / severity: critical

`contractBurned` & `contractRewarded` are not set to `0` after each burn or reward.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

7. Gas needed to process unbounded loop in depositAndStake can possibly surpass block limit

type: usability / severity: medium

`staker.contracts` array keeps growing indefinitely, possibly making adding new stakes impossible at some point.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/NexusMutual/smart-contracts/commit/f738e8da53ab1ccbf57b6ae58e6790842ac5b524

Notes

After user's stake has been burned, the `sum of all stakes <= deposit * MAX_LEVERAGE` invariant can be broken.