Losses can be frontrun in Strategy

Details

Severity - Medium
Target - https://github.com/GalloDaSballo/wbtc-levered-strat
Commit - c3feef72b7c342f690c64cb501026adc489f7c0d

Description

Smart Contract Bug Description

When a user withdraws from a yearn v2 vault, these withdrawals can, if they exceed the reserve of the vault, start withdrawing from individual strategies. The main logic for this is situated in the vault, and uses the strategies liquidatePosition function to divest assets from active strategy positions and return to the vault. Generally speaking any losses accrued through these types of withdrawal are shifted on to the user, by having the yearn strategy satisfying the the following invariant:

```
_liquidatedAmount + _loss <= _amountNeeded
```

Which is sensible, as you don't want potential loss to spill over to the rest of the vault, and by doing so, socializing the loss of the strategy to the other vault holders. The way the WBTC strategy defines loss, as only being applied if the following invariant is broken:

```
want.balanceOf(this) >= _liquidatedAmount
```

causes issues with the first invariant. As it will always be true as loss will equal 0 as long as there's enough assets to cover the current withdrawal. This can cause a situation where the strategies' aave position gets divested, but no losses realized as the users withdrawal amount is less than the total balance of the position/strategy. This means that a user could attempt to withdraw their assets from the strategy before any of its losses are realized, and thus socializing the loss to any users who remain in the vault.

We can compare a similar scenario in the generic leveraged compound strategy, where we indeed see that they don't compare withdrawal against strategy assets to determine loss, but strategy total assets vs, strategy debt.

Proof of concept/Steps to Reproduce

An attacker can frontrun any loss in the strategy by withdrawing less assets than the strategy has available after it has divest its aave position.

Pending a clean write up of the steps taken for this proof, a breakdown of this can be found in this discord thread

Impact

While individual users potentially can frontrun losses in this strategy, the circumstances for doing so are depending on one of several factors, i.e. what position the strategy has in the withdrawal queue, how much assets the strategy has compared to other strategies in the vault. A larger concern would be the potential to use this as a stepping block for a more severe attack, as the user can move out their assets from the vault without suffering any loss that has occured in the leveraged WBTC strategy.

Risk Breakdown

- Difficulty to Exploit: Easy - but currently relies on preconditions unless the user can force these through other interactions
- Severity: Medium

Recommendation

Change the loss calculations in liquidatePosition from:

```
_loss = _amountNeeded.sub(totalAssets);
```

To

```
_loss = debtOutstanding.sub(totalAssets);
```

where `debtOutstanding` => vault.debtOutstanding();

References

- GenericLevComp.sol: https://github.com/groLabs/gro-strategies/blob/main/contracts/strategies/GenericLevComp.sol#L587