JointStrategy Audit by Team2 (takeda)

General

Very well written code, some areas might benefit from some documentation.
Some booleans have a 'negative' naming, I feel using 'positive' naming makes things less confusing. (ex: allowWithdraw instead of dontWithdraw).
You could also update to a more recent version of solidity.

Provider Strategy

Nothing to report

Joint

TAR-XXX: _initialize: comment and code do not match

Tools/Techniques: Manual
Difficulty+Impact: Very minor

Details

maxPercentageLoss set to 500 although 0.1% is written in comment

Mitigation

Assuming code is correct, 0.1% should be 5%

TAR-XXX: closePositionReturnFunds: function ends if only one of the invested amount is 0

Tools/Techniques: Manual
Difficulty+Impact: Minor

Details

Though in almost all cases an invested amount of 0 means there is no invested amounts for each of the token, it may be worth handling the case where one of the token has an amount invested.

```
if (investedA == 0 || investedB == 0) {...
```

Mitigation

Pull any leftover invested amount if only one of the token is equal to 0

TAR-XXX: closePosition: possibly infinite slippage when removing liquidity

Tools/Techniques: Manual
Difficulty+Impact: Minor

Details

With an infinite slippage allowed, the strategy ensures the call almost never fails, but leaves it somewhat vulnerable to the LP's liquidity becoming imbalanced.

Mitigation

Determine an acceptable slippage for removing liquidity, while also allowing infinite slippage in case the acceptable slippage ends up locking the funds for too long

TAR-XXX: swapReward: strategy ignores the ratio if one of the token is weth

Tools/Techniques: Manual
Difficulty+Impact: Minor

Details

```
if (tokenA == WETH || tokenB == WETH) {
    return (WETH, sellCapital(reward, WETH, _rewardBal));
}
```

The strategy ignores the ratios of tokenA and tokenB if one of them is WETH, why is that ?

Mitigation

Apply the same process to the token if it is WETH as if it weren't

HegicJoint

TAR-XXX: closeHedgeManually:

Tools/Techniques: Manual
Difficulty+Impact: Minor

Details

I may misunderstand the function's prupose, but if it is called when a hedge needs to be closed no matter what, then you may want to get rid of the require statement altogether

Mitigation

Remove require statement

TAR-XXX: setHedgingPeriod: max time for a period could be replaced with a constant

Tools/Techniques: Manual
Difficulty+Impact: Very minor

Details

`require(_period < 90 days);`
Most require statements in this contract use a constant instead of the raw value, this could be applied here too.

Mitigation

Replace 90 days with a constant (ex: MAX_PERIOD);

SushiJoint

Minor improvement : replace " > 0 " checks for uint256 vars with " != 0 "

LPHedgingLib

TAR-XXX: getLPInfo: information for both of the tokens is collected, but only one is returned

Tools/Techniques: Manual
Difficulty+Impact: Minor

Details

See title

Mitigation

The mainAsset could be determined first, and then only the information of said mainAsset needs to be collected