Known Effects of Hook Permissions

Things that hooks could do that could be malicious or frustrating to users.

Note: “JIT” ⇒ “just in time” liquidity. Liquidity thats added immediately before a swap to get LP fees. Often then removed again immediately after the swap.

Hooks without Custom Accounting

beforeSwap

- cause a revert (either through pushing the price, just reverting outright, or removing liquidity)
- frontrun the user’s swap, pushing user to their max slippage
- cause a partial fill (by removing liquidity)
- JIT causing other in-range LPs to reap fewer fees

afterSwap

- cause a revert
- backrun

beforeSwap+afterSwap

- a guaranteed (risk-free) sandwich of the swap
- a guaranteed (risk-free) JIT of liquidity causing other in-range LPs to reap less reward

before addLiquidity

- cause a revert
- cause the ratio of the two tokens owed to the pool to be different than expected by swapping to different price

after addLiquidity

- cause a revert

before removeLiquidity

- cause a revert, implying user funds could be permanently locked and fees to never be collected

after removeLiquidity

- cause a revert, implying user funds could be permanently locked and fees to never be collected

before donate

- cause a revert

after donate

- cause a revert

before+after donate

- sandwich the donation and potentially capture all of it

Custom Accounting Hooks

beforeSwap returns delta

- **The below can ONLY happen if the hook also has the beforeSwap hook**
- exactInput swaps:
  - can push swapper to max slippage (ie if the router reverts when maxOutputAmount < deltaUnspecified, a hook can set deltaUnspecified to maxOutputAmount)
  - can “take” all specified token without crediting the user with anything
    - should be checked in a router
  - can take full unspecified amount (if nonzero)
    - should be checked in a router
  - On low liquidity pools, note an example for a badly written hook that blindly credits without checking liquidity status. A user can always take the full creditable amount from the hook, and in this case it happens without the user paying anything.

- lets say a hook gives the user an extra 1% of amountSpecified to every trade
- lets say the pool only has liquidity for 1 ETH → 3500 USDC left available

1. a user trades 100 ETH exact input, the hook contributes 1 ETH
2. the pool tries to trade 101 ETH exact input, but only 1 ETH liquidity is available to trade on the pool
3. 1 ETH is taken from the hook and credited to the pool manager, and the user pays (1 ETH - 1ETH) 0 ETH input, and is given all 3500 USDC output.

- exactOutput swaps:
    - can push swapper to max slippage (ie if the router reverts when maxInputAmount > deltaUnspecified revert, a hook can set deltaUnspecified to maxInputAmount - 1)
    - can take full unspecified amount, CANNOT take any of specified amount
        - should be checked in a router
    - On low liquidity pools, note this example for a badly written hook that blindly credits without checking liquidity status. A user can always take the full creditable amount from the hook.
        - lets say a hook gives the user an extra 1% of amountSpecified to every trade
        - lets say the pool only has liquidity for 1 ETH → 3500 USDC left available
        1. a user trades 700,000 USDC exactOutput, the hook gives 7,000 USDC output
        2. the pool tries to trade for 693,000 USDC output, but only 3,500 is available for 1 ETH input
        3. the user is charged 1 ETH, and is given (7,000 + 3,500) 10,500 USDC

afterSwap returns delta

- **The below can ONLY happen if the hook also has the afterSwap hook**

- can take full unspecified token from user
    - should be checked in a router

afterAddLiquidity returns delta

- **The below can ONLY happen if the hook also has the afterAddLiquidity hook**
- can take extra amounts of token0 and token1 (fee on add)
    - should be checked in a router
- can pay for the amounts of token0 and token1 on behalf of user (on add)

afterRemoveLiquidity returns delta

- **The below can ONLY happen if the hook also has the afterRemoveLiquidity hook**
- can take full amounts in both tokens from user
    - should be checked in a router