NoFeeSwap disclaimers - **core**[*]

1. **Investigation of hook contracts and permission flags:** Liquidity providers should exercise caution by carefully investigating hook contracts and permission flags of every poolId in which they intend to deposit liquidity.

2. **Just-in-time liquidity vulnerabilities:** The core's donate method is vulnerable to *just-in-time* liquidity attacks within both the consensus layer and through pre/mid/post hook mechanisms, which could be used to divert donations from liquidity providers.

3. **Position removal restrictions:** The burn hooks have the capability to prevent the removal of liquidity provider positions.

4. **Low-liquidity pool risk:** Price data obtained from low-liquidity pools lacks reliability and is susceptible to manipulation.

5. **Exchange amount precision:** The protocol does not guarantee exact matching of the amountSpecified parameter:

    - For incoming amounts (positive amountSpecified), the actual incoming amount will not exceed the specified value but approximates it as closely as possible given logPriceLimit and crossThreshold constraints. This is due to X59 granularity of target.
    - For outgoing amounts (negative amountSpecified), the actual outgoing amount will not be less than the specified value but approximates it as closely as possible given logPriceLimit and crossThreshold constraints. This is due to X59 granularity of target.
    - Any residual dust amounts due to X59 granularity of overshoot are credited to the swapper.

[*]https://github.com/NoFeeSwap/core.

6. **Hash collision vulnerability:** The protocol is vulnerable to hash collision. In the event of a keccak256 hash collision, funds can be withdrawn from the singleton.

7. **Lack of ERC-7751 compliance:** The protocol is not ERC-7751 compliant. Error messages are relayed back directly without bubbling.

8. **Token quantity limits:** The protocol does not support possession of a token amount beyond type(uint128).max. Tokens with a total supply exceeding this limit are not supported.

9. **Assembly blocks:** The "memory − safe" annotation for assembly blocks is avoided to prevent compilers from interfering with memory.

10. **Similar identifier risks:**

    - poolIds can appear visually similar, potentially deceiving users.
    - Similar-looking tags can be generated through brute-force search.

11. **tag-wrapping confusion:** If

    tag0 = Tag.wrap(uint256(uint160(ERC20Address)))

    corresponds to an ERC-20 address, then

    ```
    Tag.wrap(
      uint256(keccak256(abi.encodePacked(address(nofeeswap), tag0)))
    )
    ```

    generates a tag representing the wrapped version of tag0. This wrapping process can continue indefinitely, potentially causing confusion.

12. **Event generation optimization:** Event generation is minimized for gas efficiency, particularly for swaps. However, pool status can be mirrored off-chain by monitoring these events.

13. **Decentralization limitations:** The Sentinel contract implementation reduces protocol decentralization. The core protocol is not fully permissionless, as it has a DAO owner and is partially controlled by the Sentinel contract.

14. **Slippage protection:** Slippage checks should be implemented externally as inputs for the Operator contract or within intermediary contracts interacting with the core.

15. **Nonstandard tokens:** The protocol does not support ERC-20 tokens with non-standard implementations, including rebasing tokens, double entry point tokens, or malicious tokens. Using such tokens can introduce vulnerabilities to their associated pools.

16. **Privileged roles:** The protocol includes trusted privileged roles: the Sentinel contract and the protocol owner DAO. The Sentinel contract can modify the protocol growth portion and the maximum value for pool growth portions.

17. **Gas exhaustion by external contracts:** When calling external contracts (e.g., a hook or a token contract), all of the remaining gas is provided to the next context. Malicious or inefficient contracts may exhaust the gas supply, causing a denial of service through return bombing.