SHERLOCK

SHERLOCK SECURITY REVIEW FOR DODO

Prepared for:	DODO
Prepared by:	Sherlock
Lead Security Expert:	
Dates Audited:	December 26 - December 30, 2023
Prepared on:	February 5, 2024

Introduction

A leveraged market making solution to minimize IL and improve on liquidity management. The solution provides yield for retail LPs, higher profits for expert LPs, and better liquidity for traders.

Scope

Repository: DODOEX/dodo-v3

Branch: d3mm-no-borrow-version

Commit: 78c5cb6ea2fb1cd8ee675a1cebcaf6332faaf99a

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
1	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

404Notfound	dirk_y	alexzoid

Issue M-1: Inability to Re-add `oldToken` After Execution of `D3MakerFreeSlot.setNewTokenAndReplace()`

Source: https://github.com/sherlock-audit/2023-12-dodo-judging/issues/41

Found by

404Notfound, alexzoid, dirk_y

Summary

The `D3MakerFreeSlot` contract's `setNewTokenAndReplace()` function, when executed, permanently removes the `oldToken` information without clearing its index from `state.priceListInfo.tokenIndexMap`. This oversight prevents the re-addition of the `oldToken` using either `setNewTokenAndReplace()` or `setNewToken()` methods.

Vulnerability Detail

The `setNewTokenAndReplace()` function is designed to replace an existing token with a new token in a slot, thereby saving gas. However, this function only removes the `oldToken`'s information from `state.tokenMMInfoMap` and does not clear its index from `state.priceListInfo.tokenIndexMap`. Consequently, the system behaves as if the `oldToken` is still present, preventing its re-addition.

Impact

Once an `oldToken` is replaced, it cannot be re-introduced into the system using standard methods, potentially leading to operational inefficiencies or the need for workaround solutions.

Code Snippet

https://github.com/sherlock-audit/2023-12-dodo/blob/main/dodo-v3/contracts/DODOV3MM/D3PoolNoBorrow/D3MakerFreeSlot.sol#L30

Proof Of Concept

The test validates that after executing `setNewTokenAndReplace()`, the `oldToken` cannot be re-added due to its retained index in `state.priceListInfo.tokenIndexMap`.

Add the function below into `new-dodo-v3/test/DODOV3MM/D3MM/D3MMNoBorrow.t.sol`.

```
function testAuditReSetToken() public {

    // Setup Phase
    // Here we initialize the necessary variables and define a common error
↪   message for later use.

    bytes memory HAVE_SET_TOKEN_INFO = "D3MAKER_HAVE_SET_TOKEN_INFO";

    Types.TokenMMInfo memory tokenMMInfo;
    uint256 tokenIndex;

    // Creating a new token (token6) for testing purposes.
    MockERC20 token6 = new MockERC20("Token 6", "TK6", 18);

    // Verifying that token2 currently exists at index 2 in the token mapping.
    (tokenMMInfo, tokenIndex) =
↪   d3MakerFreeSlotWithPool.getTokenMMInfoForPool(address(token2));
    assertEq(tokenIndex, 2);

    // Confirming that the new token (token6) does not yet exist in the mapping.
    (tokenMMInfo, tokenIndex) =
↪   d3MakerFreeSlotWithPool.getTokenMMInfoForPool(address(token6));
    assertEq(tokenIndex, 0);

    // Replacing token2 with token6
    MakerTypes.TokenMMInfoWithoutCum memory token6Info = contructToken1MMInfo();
    vm.prank(maker);
    d3MakerFreeSlotWithPool.setNewTokenAndReplace(
        address(token6),
        true,
        token6Info.priceInfo,
        token6Info.amountInfo,
        token6Info.kAsk,
        token6Info.kBid,
        address(token2)
    );

    // Checking that token6 has taken over the index position of token2.
    (tokenMMInfo, tokenIndex) =
↪   d3MakerFreeSlotWithPool.getTokenMMInfoForPool(address(token6));
    assertEq(tokenIndex, 2);

    // Ensuring that token2 is no longer present in the pool.
    (tokenMMInfo, tokenIndex) =
↪   d3MakerFreeSlotWithPool.getTokenMMInfoForPool(address(token2));
```

```
    assertEq(tokenIndex, 0);

    // Confirming that token2's original index in the mapping was not cleared
↪   during the replacement.
    assertEq(d3MakerFreeSlotWithPool.getOneTokenOriginIndex(address(token2)), 2);

    // Validation Phase
    // In this section, we attempt to re-add token2 back to the pool in place of
↪   token6, expecting it to fail.

    // Attempt to replace token6 with token2 again, expecting a revert due to
↪   token2's index still being present.
    MakerTypes.TokenMMInfoWithoutCum memory token2Info = contructToken2MMInfo();
    vm.prank(maker);
    vm.expectRevert(HAVE_SET_TOKEN_INFO);
    d3MakerFreeSlotWithPool.setNewTokenAndReplace(
        address(token2),
        true,
        token2Info.priceInfo,
        token2Info.amountInfo,
        token2Info.kAsk,
        token2Info.kBid,
        address(token6)
    );

    // Attempt to add token2 as a completely new token, still expecting a revert
↪   due to its lingering index.
    vm.startPrank(maker);
    vm.expectRevert(HAVE_SET_TOKEN_INFO);
    d3MakerFreeSlotWithPool.setNewToken(
        address(token2),
        true,
        token2Info.priceInfo,
        token2Info.amountInfo,
        token2Info.kAsk,
        token2Info.kBid
    );
}
```

Run test with `forge test --match-test testAuditReSetToken`. Output example:

```
Running 1 test for test/DODOV3MM/D3MM/D3MMNoBorrow.t.sol:D3MMNoBorrowTest
[PASS] testAuditReSetToken() (gas: 691465)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 12.81ms

Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
```

Tool used

VSCode, Foundry

Recommendation

The suggested fix involves modifying the `setNewTokenAndReplace()` function to remove the `oldToken` from `state.priceListInfo.tokenIndexMap`:

```
diff --git a/new-dodo-v3/contracts/DODOV3MM/D3PoolNoBorrow/D3MakerFreeSlot.sol
 ↪  b/new-dodo-v3/contracts/DODOV3MM/D3PoolNoBorrow/D3MakerFreeSlot.sol
index 43d028d..a11b348 100644
--- a/new-dodo-v3/contracts/DODOV3MM/D3PoolNoBorrow/D3MakerFreeSlot.sol
+++ b/new-dodo-v3/contracts/DODOV3MM/D3PoolNoBorrow/D3MakerFreeSlot.sol
@@ -79,6 +79,9 @@ contract D3MakerFreeSlot is D3Maker {
         state.priceListInfo.tokenIndexMap[token] = tokenIndex + 1;
         state.tokenMMInfoMap[token].tokenIndex = uint16(tokenIndex);

+        // Remove oldToken from list
+        state.priceListInfo.tokenIndexMap[oldToken] = 0;
+
         emit SetNewToken(token);
         emit ReplaceToken(oldToken, token);
     }
```

Discussion

sherlock-admin2

1 comment(s) were left on this issue during the judging contest.

karanctf commented:

> low

nevillehuang

See also #19 for alternative described impact

nevillehuang

As confirmed by sponsor, watsons highlighted a valid scenario where re-addition of a removed token is not possible, so this would constitute breaking core contract functionality of readding tokens, so medium severity is appropriate.

traceurl

We have fixed the bug in this PR: https://github.com/DODOEX/dodo-v3/pull/8

rcstanciu

The Lead Senior Watson signed-off on the fix.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.