# Venus SwapHelper Audit Report

## Please Note

1. The analysis of the Severity is purely based on the smart contracts mentioned in the Audit Scope and does not include any other potential contracts deployed by the Owner. No applications or operations were reviewed for Severity. No product code has been reviewed.

2. Due to the time limit, the audit team did not do much in-depth research on the business logic of the project. It is more about discovering issues in the smart contracts themselves.

**Audit Period:** 2025/11/11 - 2025/10/12 (YYYY/MM/DD)

**Overall Risk:**

| | |
|---|---|
| **Project Name** | Venus SwapHelper Audit Report |
| **Github** | https://github.com/VenusProtocol/venus-periphery/pull/5 |

## Smart contracts list:

| No. | Contract Name | Type | Link | Verdict | Details |
|---|---|---|---|---|---|
| 1 | SwapHelper.sol | Helper | https://github.com/VenusProtocol/venus-periphery/blob/feat/swapper/contracts/SwapHelper/SwapHelper.sol |  | [I01] [I02] [I03] |

## Findings

| [I01] Inflexibility for token approvals | |
|---|---|
| **Contract** | SwapHelper.sol |
| **Severity Level** | Informational |
| **Description** | In the SwapHelper contract, there is an approveMax function that allows a caller with the appropriate signature or the owner to approve the maximum amount of allowance for a specific ERC20 token.<br>However, this function does not provide capabilities to:<br>● Revoke any unspent allowances<br>● Specify the exact amount of tokens to approve for the spender<br><br>JavaScript<br>`function approveMax(IERC20Upgradeable token, address spender) external onlyOwnerOrSelf {`<br>`token.forceApprove(spender, type(uint256).max);`<br>`emit ApprovedMax(address(token), spender);`<br>`}` |
| **Recommendation** | Consider including flexibility for approval amounts to allow the caller or owner to remove allowances once the swap is completed during the multicall.<br><br>If not, the caller/owner would have to do this via the genericCall which is meant to perform swaps. |
| **Status** | Acknowledged, no fix implemented. Any custom approvals required can be performed through genericCall instead when required |

| [I02] Missing zero address check in sweep Function | |
|---|---|
| **Contract** | SwapHelper.sol |
| **Severity Level** | Informational |
| **Description** | The sweep function, which transfers the contract's entire balance of a given token, does not validate that the recipient to address is not address(0). While this function is protected and can only be called by the contract owner or via a backend-signed multicall, an error could lead to address(0) being provided as the recipient. |

```
JavaScript
    function sweep(IERC20Upgradeable token, address to) external
onlyOwnerOrSelf {
        uint256 amount = token.balanceOf(address(this));
        if (amount > 0) {
            token.safeTransfer(to, amount);
        }
        emit Swept(address(token), to, amount);
    }
```

While many standard ERC20 tokens revert on transfers to the zero address, this may not be a universal standard. If a token that allows transfers to address(0) is swept, the funds would be permanently and irrecoverably burned.

| | |
|---|---|
| **Recommendation** | Consider implementing a zero-address check for the to parameter at the beginning of the sweep function |
| **Status** | Fixed as recommended, now includes a zero address check for the to variable |

## [I03] Missing comments

| | |
|---|---|
| **Contract** | SwapHelper.sol |
| **Severity Level** | Informational |
| **Description** | In the generiCall/sweep/approveMax function, it is stated that such functions should only be called via the multi-call, but misses an additional comment that the owner can call these functions to restore the contract state |
| **Recommendation** | Consider including a comment to state that the owner is capable of calling these permissioned functions |
| **Status** | Fixed, now include comprehensive comments that states the possibility of invocation by the owner role. |