Quantstamp

# Venus - SwapperHelper

## Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

| | |
|---|---|
| Type | Utility Contract |
| Timeline | 2025-11-11 through 2025-11-17 |
| Language | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | None |
| Source Code | • VenusProtocol/venus-periphery #60261ec |
| Auditors | • Ibrahim Abouzied Auditing Engineer<br>• Leonardo Passos Senior Research Engineer<br>• Hytham Farah Auditing Engineer |

| | |
|---|---|
| Documentation quality | High |
| Test quality | High |
| Total Findings | 3 Acknowledged: 3 |
| High severity findings | 0 |
| Medium severity findings | 0 |
| Low severity findings | 2 Acknowledged: 2 |
| Undetermined severity findings | 0 |
| Informational findings | 1 Acknowledged: 1 |

## Summary of Findings

This audit covers the `SwapHelper` contract, a periphery contract in the Venus Protocol. The `SwapHelper` contract helps Venus client contracts interpret and execute token swap routes generated by the SwapAPI. This is done with a simple multi-call with built-in replay protection and verification that the multi-call payload was signed by the SwapAPI. The upstream Venus client contracts are responsible for funding the swap and validating that they have received the expected token amounts at the end of the swap.

Over the course of the audit, we identified some low severity vulnerabilities. `genericCall` currently allows arbitrary external interactions, posing risks if the `backendSigner`'s key is compromised (VEN-1). The use of `approveMax()` also raises concerns about unlimited approvals, exposing users to potential risks if trust is misplaced (VEN-2). While the contract is simple and most of the attack surface lies off-chain or in integrating/downstream contracts, addressing these vulnerabilities can help strengthen the security of the entire system.

**Fix-Review Update**: The Venus team has acknowledged vulnerabilities VEN-1, VEN-2, and VEN-3, providing explanations regarding the intended functionality of the `SwapHelper` contract.

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| VEN-1 | `genericCall()` Allows Arbitrary External Calls | • Low | Acknowledged |
| VEN-2 | Max Approval Can Put Funds At Risk | • Low | Acknowledged |
| VEN-3 | Non-Atomic Pre-Funding Can Be Front-Run | • Informational | Acknowledged |

## Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
>
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

**Possible issues we looked for included (but are not limited to):**

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

**Methodology**

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

# Scope

The audit was done on the pull request [VPD-71]: Add generic swapper contract.

The scope was the files:

- `contracts/SwapHelper/SwapHelper.sol`

# Operational Considerations

- Any SwapAPI `multicall` should be considered publicly accessible for anyone to duplicate on-chain. While the contract validates that `multicall`s from the SwapAPI cannot be replayed within it, it is still possible for a user to make a duplicate call with a contract of their own. This is because the `SwapHelper` operates as a helper contract for interpreting SwapAPI data and doesn't possess any privileged roles in downstream contracts.
- SwapHelper must be pre-funded with the relevant tokens before a multicall executes, because it relies on existing balances rather than pulling assets from users.
- There is no on-chain emergency pause. If the backend signer is compromised, the owner must rotate keys on-chain while operations remain live, so detection and incident response processes are critical.
- The multicall deadline check uses `block.timestamp > deadline`, allowing execution in the same block as the deadline. Operators should account for this extra block when setting expiration thresholds.
- The contract assumes fee-on-transfer tokens are not involved; `sweep` transfers `balanceOf(address(this))` without adjusting for transfer fees, which could lead to shortfalls.
- The EIP-712 domain separator is fixed to the deployment chain ID, preventing cross-chain signature replays but requiring explicit signer management on each chain.
- Multicall executes external calls serially. State changes between calls (e.g., market movements between approve and swap) can affect downstream steps and must be handled off-chain.
- The `usedSalts` mapping grows monotonically. Long-term operation should budget for incremental storage growth or plan for future compaction strategies.
- The `backSigner` and contract owner are trusted; rely on security best practices to protect their private keys.
- If a Swap Client makes a multicall swap (`approveMax`, `swap`, `sweep`), it should check that the amount received is the expected one. In case something fails, the `owner` can always perform a `sweep` in case the `SwapHelper` inadvertently holds funds, fixing corresponding balances.

- When calling `genericCall(target, data)`, the validation of the `target` and `data` occurs off-chain.
- `SwapHelper` does not support native token swaps. It is the client contract's responsibility to wrap native tokens.

# Key Actors And Their Capabilities

## Roles

### Owner

- Manages the backend signer address that authorizes multicall operations
- Directly executes administrative functions without signature requirements

**Backend Signer (SwapAPI):**

- Signs and authorizes user-initiated multicall operations through EIP-712 signatures
- Acts as the off-chain authorization layer for complex token operations
- Validates parameters, prevent malicious operations, and enforce business logic before signing
- Manages replay protection through unique salt values for each operation
- Enforces deadline timestamps to prevent delayed execution

## Exclusive Functions

All contract self-calls must be signed by the `backendSigner`.

The contract owner or the contract itself (via `backendSigner` signatures) can call:

- `genericCall()`: Calls any target contract with any call data.
- `sweep()`: Send the full token balance of any token to a target address.
- `approveMax()`: Delegates the maximum token approval for the contract to any address.

Only the contract owner can call:

- `setBackendSigner()`: Assigns the address that must sign `multicall` payloads.

# Findings

## VEN-1 `genericCall()` Allows Arbitrary External Calls

Low | Acknowledged

**Update**

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Contracts that would be clients of the  SwapHelper  contract are responsible
  for validating effects of call to  SwapHelper.multicall  or  SwapHelper.genericCall  if
  they do not receive funds they should revert the transactions.
```

**File(s) affected:** `contracts/SwapHelper/SwapHelper.sol`

**Description:** `genericCall()` permits the owner or contract (via signed multicall) to execute arbitrary external calls without target validation. If the `backendSigner`'s private key is compromised, the `owner` has to set a new one in the `SwapHelper` contract.

Until one notices the private key has been compromised, the malicious `backendSigner` can submit many multicall requests, including those with a `genericCall`. Since the latter does not enforce which target addresses are allowed, funds can be at risk.

**Recommendation:** Rely on a owner-controlled whitelist of target addresses that a `genericCall` can invoke. This gives the system an extra layer of security in case the `backendSigner` is compromised.

## VEN-2 Max Approval Can Put Funds At Risk

Low | Acknowledged

**Update**

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
SwapHelper  contract is not intended to hold any funds after transactions.
  approveMax(IERC20Upgradeable token, address spender)  function is only meant to
 be used as a helper for Swap API  multicall  builder, which might skip this step
 in case when  SwapHelper  would have enough approval to perform swap.
 In case when we would like to revoke a contract (for example when we would
```

```
like to not use certain protocol within our Swap API), owner can call
genericCall and set it to 0 .
```

**Description:** The happy flow of a multicall transaction is given by `approveMax -> genericCall (swap) -> sweep` for a given `tokenIn` and `tokenOut` pairs. Under this flow, the `SwapHelper` balance for any token is expected to be zero. However, depending on other use cases or combinations of calls, funds could be sent to the `SwapHelper` contract.

Additionally, the function `approveMax()` grants spenders an infinite allowance. If a previously trusted spender becomes malicious or compromised, the contract's existing balance and future deposits are exposed without further interaction from the owner.

**Recommendation:** Instead of giving a full approval and assuming traded tokens are not fee-on-transfer, replace the `approveMax` function with an `approve(amount)` version. Then, multicalls from the Swap API should call it to match the exact amount of tokens to be swapped.

## VEN-3 Non-Atomic Pre-Funding Can Be Front-Run

Informational | Acknowledged

**Update**

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
SwapHelper contract is not intended to hold any funds after transactions.
It should be used only through the client contracts that send funds to it in the
same transaction before calling SwapHelper.multicall .
```

**File(s) affected:** `contracts/SwapHelper/SwapHelper.sol`

**Description:** Users are required to pre-fund the `SwapHelper` contract before calling `SwapHelper.multicall()`. However, if this is not done within the same transaction, another caller can use the tokens as funds for their own `multicall` and claim the tokens in a `sweep()` call.

**Recommendation:** Have the `SwapHelper` call `transferFrom()` to transfer the funds from the users, or update the SwapAPI to include a `transferFrom()` call.

# Auditor Suggestions

## S1 Missing Address Validation in `setBackendSigner()`

Acknowledged

**Update**

Marked as "Acknowledged" by the client.

**File(s) affected:** `contracts/SwapHelper/SwapHelper.sol`

**Description:** `setBackendSigner()` rejects the zero address but allows re-setting the same signer or pointing to the contract's own address, which can emit confusing events or render signature flows unusable.

**Recommendation:** Disallow setting the signer to the current value or the contract itself, and optionally enforce EOA validation if required by operations.

## S2 Critical Role Transfer Not Following Two-Step Pattern

Fixed

**Update**

Marked as "Fixed" by the client.
Addressed in: `74ab88ce7f0cbd54c89b92548eadc91798de0c2b` .

**File(s) affected:** `contracts/SwapHelper/SwapHelper.sol`

**Description:** The owner of the contracts can call `transferOwnership()` to transfer the ownership to a new address. If an uncontrollable address is accidentally provided as the new owner address then the contract will no longer have an active owner, and functions with the `onlyOwner` modifier can no longer be executed.

**Recommendation:** Consider using OpenZeppelin's `Ownable2Step` contract to adopt a two-step ownership pattern in which the new owner must accept their position before the transfer is complete.

## S3 Ownership Can Be Renounced

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.

**File(s) affected:** `contracts/SwapHelper/SwapHelper.sol`

**Description:** Ownership can be renounced. We recommend overriding the `renounceOwnership()` to prevent losing any access to owner-required functions.

**Recommendation:** Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

# Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not pose an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

# Appendix

### File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

### Files

- `27a...4dc ./SwapHelper/SwapHelper.sol`

### Tests

- `b41...6a2 ./SwapHelper/SwapHelper.ts`

# Test Suite Results

The test suite was ran with the command `yarn test tests/hardhat/SwapHelper/SwapHelper.ts`.

```
SwapHelper
  constructor
    ✔ should revert when backendSigner is zero address
  ownership
    ✔ should set initial owner correctly
    ✔ should allow owner to transfer ownership
    ✔ should prevent non-owner from transferring ownership
  setBackendSigner
    ✔ should allow owner to change backend signer
    ✔ should prevent non-owner from changing backend signer
```

```
    ✔ should revert when setting zero address as backend signer
  sweep
    ✔ should sweep ERC20 tokens to specified address
    ✔ should revert if called by non-owner outside multicall
    ✔ should work within multicall
    ✔ should handle sweep when balance is zero
    ✔ should emit Swept event
    ✔ should emit Swept event with zero amount when balance is zero
  approveMax
    ✔ should approve maximum amount to a spender
    ✔ should emit ApprovedMax event
    ✔ should revert if called by non-owner outside multicall
    ✔ should work within multicall
  multicall
    ✔ should revert if calls array is empty
    ✔ should emit MulticallExecuted event
    ✔ should emit MulticallExecuted with correct call count
    ✔ should revert if deadline is in the past
    ✔ should check signature if provided
    ✔ should revert if the signature is invalid
    ✔ should revert if salt is reused


  24 passing (2s)
```

# Code Coverage

Code coverage was gathered by running the command `npx hardhat coverage --testfiles "tests/hardhat/SwapHelper/SwapHelper.ts"`. The test suite shows strong coverage, though we encourage the Venus team to reach 100%.

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|---|---|---|---|---|---|
| contracts/SwapHelper/ | 92.31 | 83.33 | 87.5 | 90 | |
| SwapHelper.sol | 92.31 | 83.33 | 87.5 | 90 | 146,162,180,181 |

# Changelog

- 2025-11-17 - Initial report
- 2025-11-20 - Final report

# About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora

- Academic institutions: National University of Singapore, MIT

**Timeliness of content**

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

**Notice of confidentiality**

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

**Links to other websites**

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

**Disclaimer**

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.