Report

v. 1.0

Customer

WheelX

Smart Contract Audit

WheelX Contracts

3rd September 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog **4**

2 Introduction **5**

3 Project scope **6**

4 Methodology **7**

5 Our findings **8**

6 Moderate Issues **9**
CVF-1. INFO 9
CVF-2. INFO 9
CVF-3. FIXED 9
CVF-4. FIXED 10
CVF-5. INFO 10

7 Recommendations **11**
CVF-6. FIXED 11
CVF-7. INFO 11
CVF-8. INFO 11
CVF-9. INFO 12
CVF-10. INFO 12
CVF-11. INFO 12
CVF-12. FIXED 13
CVF-13. FIXED 13
CVF-14. FIXED 13
CVF-15. FIXED 13
CVF-16. FIXED 14
CVF-17. INFO 14
CVF-18. FIXED 14
CVF-19. INFO 14
CVF-20. FIXED 15
CVF-21. INFO 15
CVF-22. FIXED 15
CVF-23. FIXED 15
CVF-24. INFO 16
CVF-25. FIXED 16
CVF-26. FIXED 16
CVF-27. FIXED 16
CVF-28. INFO 17
CVF-29. INFO 17
CVF-30. INFO 17
CVF-31. FIXED 17

CVF-32. INFO 18
CVF-33. INFO 18
CVF-34. FIXED 18
CVF-35. FIXED 18
CVF-36. INFO 19
CVF-37. FIXED 19
CVF-38. FIXED 19
CVF-39. FIXED 19
CVF-40. FIXED 20
CVF-41. FIXED 20
CVF-42. FIXED 20
CVF-43. INFO 20
CVF-44. FIXED 21
CVF-45. FIXED 21
CVF-46. FIXED 21

1 Changelog

#	Date	Author	Description
0.1	03.09.25	A. Zveryanskaya	Initial Draft
0.2	03.09.25	A. Zveryanskaya	Minor revision
1.0	03.09.25	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

WheelX is built with a mission to democratize access to decentralized liquidity, streamline the complexity of cross-chain operations, and empower users with greater flexibility.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	ApprovalProxy.sol	Multicall3Router.sol	WheelxReceiver.sol
src/libs			
	IRouter.sol		

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

No critical issues have been identified.
We provided the client with several recommendations.

Moderate	Info	Fixed
	3	2

6 Moderate Issues

CVF-1 INFO

- **Category** Procedural
- **Source** ApprovalProxy.sol

Description The error message from a failed call is dropped here, which could make error investigations harder.

Recommendation Include the original error message into the error as a parameter.

Client Comment *It's intended to save gas.*

```
72 if (!success) {
       revert NativeTransferFailed();
```

CVF-2 INFO

- **Category** Suboptimal
- **Source** Multicall3Router.sol

Description Using the free memory pointer here is redundant, as the code reverts anyway.

Recommendation Just copy the returned data into the memory starting at zero address.

Client Comment *This is for the edge case when approve return '0' without panic.*

```
59 let ptr := and(0xffffffffffffffffffffffff, mload(0x40))
60 returndatacopy(ptr, 0x00, returndatasize())
   revert(ptr, returndatasize())
```

CVF-3 FIXED

- **Category** Unclear behavior
- **Source** Multicall3Router.sol

Description The returned avlue is silently ignored.

Recommendation Explicitly require the returned value to be true.

Client Comment *Changed to use safeTransfer.*

```
179 IERC20(token).transfer(refundTo, balance);
```

CVF-4 FIXED

- **Category** Procedural
- **Source** WheelxReceiver.sol

Description The index of the failed call and the error message returned from it are dropped here, which could make error investigations harder.

Recommendation Include the index of the failed call and the error message into the error as parameters.

Client Comment *Added error reason to event.*

```
54 if (!success) {
       revert CallFailed();
```

CVF-5 INFO

- **Category** Procedural
- **Source** WheelxReceiver.sol

Description The error message of a failed call is dropped here, which could make error investigations harder.

Recommendation Include the original error message into the error as a parameter.

Client Comment *Same as 1, It's intended to save gas.*

```
78 if (!success) {
       revert NativeTransferFailed();
```

7 Recommendations

CVF-6 FIXED

- **Category** Procedural
- **Source** IRouter.sol

Recommendation The “refundTo” argument should be declared as “payable”.

```
5 function safeMultiCall(bytes calldata calls, address refundTo,
    ↪ bytes32 request_id) external payable returns (bytes memory
    ↪ returnData);
```

CVF-7 INFO

- **Category** Documentation
- **Source** IRouter.sol

Description The format of the “calls” argument is unclear.

Recommendation Explain in a documentation comment.

Client Comment *It’s intended to make it generic.*

```
5 function safeMultiCall(bytes calldata calls, address refundTo,
    ↪ bytes32 request_id) external payable returns (bytes memory
    ↪ returnData);
```

CVF-8 INFO

- **Category** Documentation
- **Source** IRouter.sol

Description The format of the “returnData” returned value is unclear.

Recommendation Explain in a documentation comment.

Client Comment *It’s intended to make it generic.*

```
5 function safeMultiCall(bytes calldata calls, address refundTo,
    ↪ bytes32 request_id) external payable returns (bytes memory
    ↪ returnData);
```

CVF-9 INFO

- **Category** Procedural
- **Source** ApprovalProxy.sol

Description We didn't review these files.

Client Comment *Come from reputed third-party.*

```
4 import {Ownable} from "solady/auth/Ownable.sol";
  import {SafeTransferLib} from "solady/utils/SafeTransferLib.sol";
```

CVF-10 INFO

- **Category** Procedural
- **Source** ApprovalProxy.sol

Recommendation The "newRouter" parameter should be indexed.

Client Comment *Should be fine, indexer can decode from log.data,and it should be rarely used.*

```
11 event RouterUpdated(address newRouter);
```

CVF-11 INFO

- **Category** Procedural
- **Source** ApprovalProxy.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *The error signature itself is also informative.*

```
12 error ArrayLengthsMismatch();
```

```
14 error NativeTransferFailed();
```

CVF-12 FIXED

- **Category** Bad datatype
- **Source** ApprovalProxy.sol

Recommendation The type for this variable should be “IRouter”.

```
16 address public router;
```

CVF-13 FIXED

- **Category** Bad datatype
- **Source** ApprovalProxy.sol

Recommendation The type for the “_router” argument should be “IRouter”.

```
18 constructor(address _owner, address _router) {
```

CVF-14 FIXED

- **Category** Procedural
- **Source** ApprovalProxy.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
24 receive() external payable {}
```

CVF-15 FIXED

- **Category** Unclear behavior
- **Source** ApprovalProxy.sol

Description This function should return the amount withdrawn.

```
26 function withdraw() external onlyOwner {
```

CVF-16 FIXED

- **Category** Bad datatype
- **Source** ApprovalProxy.sol

Recommendation The argument type should be “IRouter”.

```
30 function setRouter(address _router) external onlyOwner {
```

CVF-17 INFO

- **Category** Unclear behavior
- **Source** ApprovalProxy.sol

Description This event is emitted even if nothing actually changed.

Client Comment *It signals the the stored router address is updated.*

```
33 emit RouterUpdated(_router);
```

CVF-18 FIXED

- **Category** Suboptimal
- **Source** ApprovalProxy.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
37 address[] calldata tokens,
   uint256[] calldata amounts,
```

CVF-19 INFO

- **Category** Documentation
- **Source** ApprovalProxy.sol

Description The data format for this argument is unclear.

Recommendation Explain in a documentation comment.

Client Comment *It’s intended to be generic.*

```
39 bytes calldata calls,
```

CVF-20 FIXED

- **Category** Procedural
- **Source** ApprovalProxy.sol

Recommendation This argument should be declared as “payable”.

```
40 address refundTo,
```

CVF-21 INFO

- **Category** Unclear behavior
- **Source** ApprovalProxy.sol

Description The data format of the returned value is unclear.

Client Comment *It’s intended to be generic.*

```
42 ) external payable returns (bytes memory) {
```

CVF-22 FIXED

- **Category** Procedural
- **Source** ApprovalProxy.sol

Recommendation Inner brackets are redundant.

```
44 if ((tokens.length != amounts.length)) {
```

CVF-23 FIXED

- **Category** Bad datatype
- **Source** ApprovalProxy.sol

Recommendation The particular gas limit should be a named constant.

```
66 // Provide at most 100k gas to the internal call, which is
```

```
69 success := call(100000, to, value, 0, 0, 0, 0)
```

CVF-24 INFO

- **Category** Procedural
- **Source** Multicall3Router.sol

Description We didn't review this file.

Client Comment *Come from reputed third-party.*

```
5 import {SafeTransferLib} from "solady/utils/SafeTransferLib.sol";
```

CVF-25 FIXED

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation This library should be moved into a separate file named "Panic.sol".

```
9 library Panic {
```

CVF-26 FIXED

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation This library should be moved into a separate file named "Revert.sol".

```
31 library Revert {
```

CVF-27 FIXED

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation This library should be moved into a separate file named "SafeApproveLib.sol".

```
46 library SafeApproveLib {
```

CVF-28 INFO

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *The error name should be informative enough.*

```
93 error InvalidOffset();
   error InvalidTarget();
   error ArrayLengthsMismatch();
```

CVF-29 INFO

- **Category** Bad datatype
- **Source** Multicall3Router.sol

Recommendation The type for this variable should be more specific.

Client Comment *It's intended to be generic.*

```
99 address public immutable multicall3;
```

CVF-30 INFO

- **Category** Bad datatype
- **Source** Multicall3Router.sol

Recommendation The type for the "_multicall3" argument should be more specific.

Client Comment *It's intended to be generic.*

```
103 constructor(address _multicall3, IPermit2 _permit2) {
```

CVF-31 FIXED

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
109 receive() external payable {}
```

CVF-32 INFO

- **Category** Documentation
- **Source** Multicall3Router.sol

Description The data format for this argument in unclear.

Recommendation Explain in the documentation comment.

Client Comment *It's intended to be generic.*

```
113 bytes calldata calls,
```

CVF-33 INFO

- **Category** Documentation
- **Source** Multicall3Router.sol

Description The data format for the returned value is unclear.

Recommendation Explain in a documentation comment.

Client Comment *It's intended to be generic.*

```
116 ) external payable returns (bytes memory) {
```

CVF-34 FIXED

- **Category** Suboptimal
- **Source** Multicall3Router.sol

Description This variable is redundant.

Recommendation Just use "address(this).balance" instead.

```
126 uint256 amount = address(this).balance;
```

CVF-35 FIXED

- **Category** Bad datatype
- **Source** Multicall3Router.sol

Recommendation The type for the "settToken" argument should be "IERC20".

```
134 function sellToPool(address sellToken, uint256 bps, address pool,
      ↪ uint256 offset, bytes memory data) external {
```

CVF-36 INFO

- **Category** Bad datatype
- **Source** Multicall3Router.sol

Recommendation The type for the “pool” argument should be more specific.

Client Comment *It’s intended to be generic.*

```
134 function sellToPool(address sellToken, uint256 bps, address pool,
    ↪ uint256 offset, bytes memory data) external {
```

CVF-37 FIXED

- **Category** Procedural
- **Source** Multicall3Router.sol

Recommendation Brackets around multiplication are redundant.

```
140 value = (address(this).balance * bps) / BASIS;
```

```
155 uint256 amount = (IERC20(sellToken).balanceOf(address(this)) * bps)
    ↪ / BASIS;
```

CVF-38 FIXED

- **Category** Bad datatype
- **Source** Multicall3Router.sol

Recommendation The type for the “token” argument should be “IERC20”.

```
173 function cleanupERC20(address token, address refundTo) external {
```

```
183 function safeApproveIfBelow(address token, address spender, uint256
    ↪ amount) external {
```

```
187 function safeApprovePermit2(address token, address spender, uint256
    ↪ amount) external {
```

CVF-39 FIXED

- **Category** Unclear behavior
- **Source** Multicall3Router.sol

Recommendation This function should return the acount transferred.

```
173 function cleanupERC20(address token, address refundTo) external {
```

CVF-40 FIXED

- **Category** Suboptimal
- **Source** Multicall3Router.sol

Recommendation It would be more efficient to use a dingle array of structs with three fields, rather than three parallel arrays. This would also make the length check unnecessary.

```
197 address[] calldata tokens,
    address[] calldata recipients,
    uint256[] calldata amounts
```

CVF-41 FIXED

- **Category** Procedural
- **Source** WheelxReceiver.sol

Description A structure declaration comes right before the “Errors” comment.

Recommendation Rearrange the code.

```
5 // --- Errors ---
```

```
7 struct Call {
```

CVF-42 FIXED

- **Category** Procedural
- **Source** WheelxReceiver.sol

Recommendation This field should be declared as “payable”.

```
8 address to;
```

CVF-43 INFO

- **Category** Procedural
- **Source** WheelxReceiver.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *The error name should be informative enough.*

```
13 error CallFailed();
   error Unauthorized();
   error NativeTransferFailed();
```

CVF-44 FIXED

- **Category** Procedural
- **Source** WheelxReceiver.sol

Recommendation This variable should be declared as “payable”.

```
23 address private immutable SOLVER;
```

CVF-45 FIXED

- **Category** Unclear behavior
- **Source** WheelxReceiver.sol

Description There is no length check for the “data” argument.

Recommendation Implement an appropriate check.

```
62 function to_bytes32(bytes memory data) internal pure returns (
   ↪ bytes32 converted) {
```

CVF-46 FIXED

- **Category** Bad datatype
- **Source** WheelxReceiver.sol

Recommendation The particular gas limit should be a named constant.

```
72 // Provide at most 100k gas to the internal call, which is
```

```
75 success := call(100000, to, value, 0, 0, 0, 0)
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting