AstraSec

Orbs Spot Security Audit Report

September 26, 2025

Contents

1 Introduction

1.1 About Orbs Spot

Orbs Spot is a decentralized trading platform offering Limit Orders for precise trades with price protection, TWAP Orders for splitting trades over time, Stop-Loss/Take-Profit orders triggered by price thresholds, and Composable Execution to combine these with custom exchange adapters for flexible, secure trading.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/orbs-network/spot.git
- CommitID: 542e1b3

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/orbs-network/spot.git
- CommitID: 7502727

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the Orbs Spot protocol. Throughout this audit, we identified a total of 3 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	–	–	–	–
High	–	–	–	–
Medium	–	–	–	–
Low	–	–	–	–
Informational	3	–	–	3
Undetermined	–	–	–	–

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~I-1~~	Improved Logic of OrderValidationLib::validate()
~~I-2~~	Improved Logic of DefaultDexAdapter::delegateSwap()
~~I-3~~	Improved Logic of SurplusLib::distribute()

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Acknowledged
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

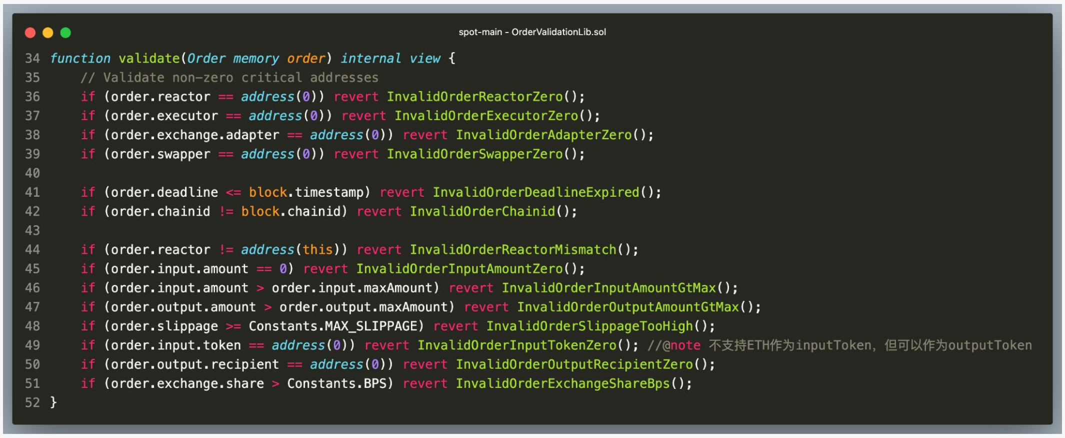
3.3 Vulnerability Details

3.3.1 [I-1] Improved Logic of OrderValidationLib::validate()

Target	Category	IMPACT	LIKELIHOOD	STATUS
OrderValidationLib.sol	Coding Practice	NA	NA	Addressed

The validate() function validates an order's structure and executability by checking addresses, reactor binding, timing/chain constraints, amounts, slippage, tokens, etc. It can be optimized by:

- ➔ Remove the redundant zero-address check for order.reactor (already excluded by order.reactor != address(this)).
- ➔ Revert when order.input.token == order.output.token to forbid self-swaps/no-ops.



spot-main - OrderValidationLib.sol

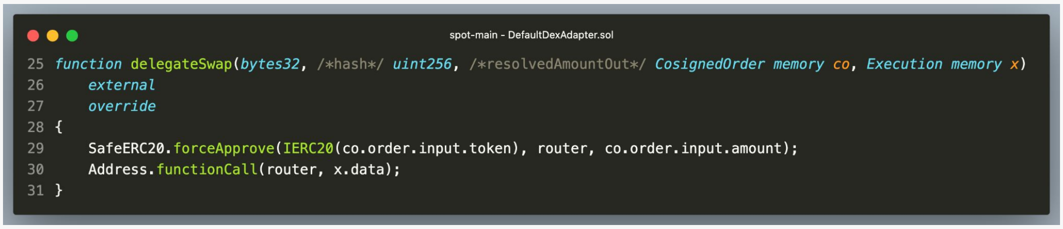
```solidity
function validate(Order memory order) internal view {
    // Validate non-zero critical addresses
    if (order.reactor == address(0)) revert InvalidOrderReactorZero();
    if (order.executor == address(0)) revert InvalidOrderExecutorZero();
    if (order.exchange.adapter == address(0)) revert InvalidOrderAdapterZero();
    if (order.swapper == address(0)) revert InvalidOrderSwapperZero();

    if (order.deadline <= block.timestamp) revert InvalidOrderDeadlineExpired();
    if (order.chainid != block.chainid) revert InvalidOrderChainid();

    if (order.reactor != address(this)) revert InvalidOrderReactorMismatch();
    if (order.input.amount == 0) revert InvalidOrderInputAmountZero();
    if (order.input.amount > order.input.maxAmount) revert InvalidOrderInputAmountGtMax();
    if (order.output.amount > order.output.maxAmount) revert InvalidOrderOutputAmountGtMax();
    if (order.slippage >= Constants.MAX_SLIPPAGE) revert InvalidOrderSlippageTooHigh();
    if (order.input.token == address(0)) revert InvalidOrderInputTokenZero(); //@note 不支持ETH作为inputToken，但可以作为outputToken
    if (order.output.recipient == address(0)) revert InvalidOrderOutputRecipientZero();
    if (order.exchange.share > Constants.BPS) revert InvalidOrderExchangeShareBps();
}
```

Remediation Optimize the validate() function as described above.

3.3.2 [I-2] Improved Logic of DefaultDexAdapter::delegateSwap()

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
DefaultDexAdapter.sol	Coding Practice	NA	NA	Addressed

In the delegateSwap() function, the adapter grants the router an allowance equal to co.order.input.amount via forceApprove(), but the allowance is not cleared after the swap. Because the adapter is executed via delegatecall in the Executor contract, this approval is given from the Executor contract to the router and may persist across fills. If the router is compromised, upgraded, or later called with stale calldata, the leftover allowance could be abused.



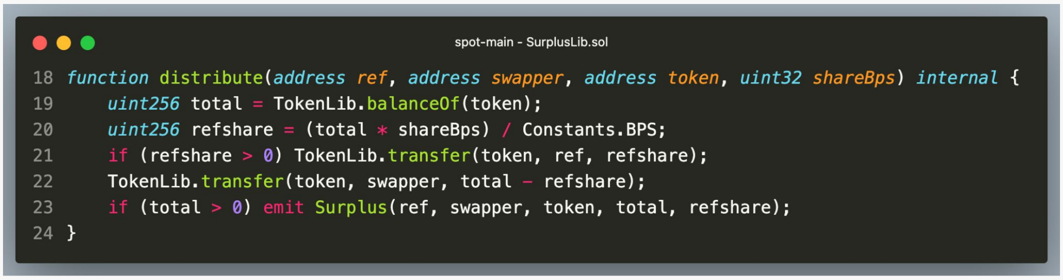
```
function delegateSwap(bytes32, /*hash*/ uint256, /*resolvedAmountOut*/ CosignedOrder memory co, Execution memory x)
    external
    override
{
    SafeERC20.forceApprove(IERC20(co.order.input.token), router, co.order.input.amount);
    Address.functionCall(router, x.data);
}
```

Remediation Reset the allowance for the router to zero right after the swap.

3.3.3 [I-3] Improved Logic of SurplusLib::distribute()

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
SurplusLib.sol	Coding Practice	NA	NA	Addressed

The distribute() function distributes surplus tokens between a referrer and a swapper based on a specified share percentage. It can be enhanced by shifting the total balance check to the start, enabling early termination if total is zero, thus avoiding unnecessary computations and improving efficiency.



spot-main - SurplusLib.sol

```solidity
function distribute(address ref, address swapper, address token, uint32 shareBps) internal {
    uint256 total = TokenLib.balanceOf(token);
    uint256 refshare = (total * shareBps) / Constants.BPS;
    if (refshare > 0) TokenLib.transfer(token, ref, refshare);
    TokenLib.transfer(token, swapper, total - refshare);
    if (total > 0) emit Surplus(ref, swapper, token, total, refshare);
}
```

Remediation Improve the distribute() function as described above.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI