Zellic

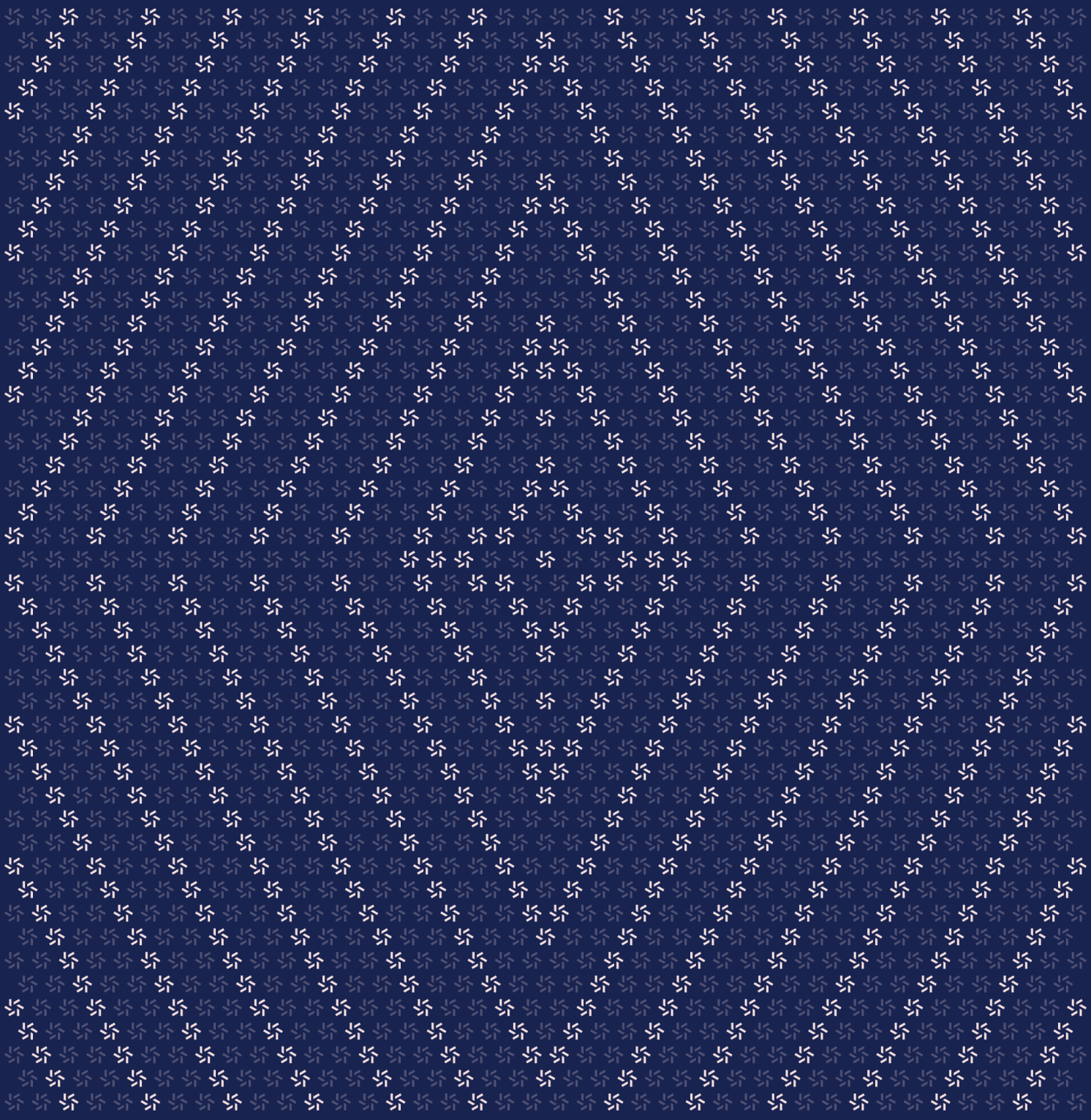
November 12, 2024

Prepared for
Matt Deible
Odos

Prepared by
Katerina Belotskaia
Sunwoo Hwang
Sylvain Pelissier
Varun Verma
Zellic

Odos Router V3

Smart Contract Patch Review



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security patch review for the Odos team on November 5th, 2024 as well as from May 22nd to May 23rd, 2025. During these engagements, Zellic reviewed Odos Router V3's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In these assessments, we sought to answer the following question:

- Were any bugs introduced during recent updates?

We were asked to review several patches to the Odos Router V3 contracts related to the referral system, which has now been moved off the chain. The purpose of this review (May 22–23, 2025) was to focus exclusively on changes to the Odos Router V3 contracts since our last review (November 5, 2024), evaluating them for any potential security vulnerabilities or inconsistencies. In sections 4. ↗ and 5. ↗, we provide overviews of the changes from each review.

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

2. Introduction

2.1. About Odos Router V3

Odos contributed the following description of Odos Router V3:

> The Odos Router serves as a security wrapper around the arbitrary execution of contract interactions to facilitate efficient token swaps.

2.2. Scope

The engagement involved a review of the following targets:

Odos Router V3 Contracts

Type	Solidity
Platform	EVM-compatible
Target	odos-router-v3
Repository	https://github.com/odos-xyz/odos-router-v3 ↗
Version	`f90fd6230e7dbdc2a71c4e12df57b2dfdeacb7ed`

Programs	`contracts/OdosRouterV3.sol`
	`contracts/OdosTransferHook.sol`
	`contracts/OdosWETHExecutor.sol`

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io

Chad McDonald
Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Katerina Belotskaia
Engineer
kate@zellic.io

Sunwoo Hwang
Engineer
sunwoo@zellic.io

Sylvain Pelissier
Engineer
sylvain@zellic.io

Varun Verma
Engineer
varun@zellic.io

2.3. Project Timeline

The key dates of the engagement are detailed below.

November 5, 2024	Kick-off call
November 5, 2024	Start of primary review of diff between `08bb5108` and `d14d55fa`
November 5, 2024	End of primary review period
May 22, 2025	Start of secondary review of diff between `d14d55fa` and `f90fd623`
May 23, 2025	End of secondary review period

3. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment(s). These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

3.1. The `fee` and `feeRecipient` can be arbitrarily changed by users

> **Note:** This observation is from our primary review period on November 5th.

The previous implementation of the OdosRouterV3 contract obtained all necessary reference information from the ReferralRegistry contract. Any user could register a new referral code along with additional information, such as the fee percentage and beneficiary address. The referral code was guaranteed to be unique and immutable, along with the data associated with it.

In the current implementation of the OdosRouterV3 contract, the management of referral information has been moved off chain. As a result, the `swapCompact`, `swap`, `swapPermit2`, `swapMultiCompact`, `swapMulti`, and `swapMultiPermit2` functions now accept referral data directly as an input argument.

```
function swapMulti(
  inputTokenInfo[] memory inputs,
  outputTokenInfo[] memory outputs,
  bytes calldata pathDefinition,
  address executor,
uint32 referralCode
swapReferralInfo memory referralInfo
) [...]
```

This input data is defined by the `swapReferralInfo` structure and includes `code`, `fee`, and the `feeRecipient` address. This change allows users performing swap actions to have full control over the referral data and to provide an arbitrary `fee` and `feeRecipient` address not defined by the referrer.

```
struct swapReferralInfo {
    uint64 code;
    uint64 fee;
    address feeRecipient;
}
```

4. Patch Review 1

This section documents notable and minor changes applied to the in-scope code from commit `08bb5108` to commit `d14d55fa` for our review on November 5th, 2024.

4.1. Notable changes

The following were notable changes made to the codebase.

ReferralRegistry contract

The ReferralRegistry contract provided functionality to register a new referral code and add information about `referralFee`, `beneficiary`, and `managers`. However, since the referral-management logic has been moved off chain, this contract has been deleted as it is no longer necessary.

OdosRouterV3 contract

The `swapCompact`, `swap`, `swapPermit2`, `swapMultiCompact`, `swapMulti`, and `swapMultiPermit2` now accept the data structure `swapReferralInfo` instead of the `referralCode` value.

```
struct swapReferralInfo {
    uint64 code;
    uint64 fee;
    address feeRecipient;
}
```

The `swapCompact` and `swapMultiCompact` functions have been modified to correctly decode the provided `swapReferralInfo` data from the calldata.

```
function swapMulti(
  inputTokenInfo[] memory inputs,
  outputTokenInfo[] memory outputs,
  bytes calldata pathDefinition,
  address executor,
uint32 referralCode
swapReferralInfo memory referralInfo
) [...]
```

The `_swap` and `_swapMulti` have been modified to include additional require statements that verify the validity of the provided referral information: the `feeRecipient` should not be zero address, and the fee cannot exceed `FEE_DENOM / 50`.

```
if (referralInfo.fee > 0) {
    require(referralInfo.feeRecipient != address(0), "Null fee recipient");
```

```
    require(referralInfo.fee <= FEE_DENOM / 50, "Fee too high");
```

4.2. Minor differences

The following were minor changes made to the codebase.

OdosRouterV3 contract

The `Swap` and `SwapMulti` events have been updated to include new fields: `referralCode`, `referralFee`, and `referralFeeRecipient`.

```
event Swap(
    address sender,
    uint256 inputAmount,
    address inputToken,
    uint256 amountOut,
    address outputToken,
    int256 slippage,
    uint32 referralCode
    uint64 referralCode,
    uint64 referralFee,
    address referralFeeRecipient
);

event SwapMulti(
    address sender,
    uint256[] amountsIn,
    address[] tokensIn,
    uint256[] amountsOut,
    address[] tokensOut,
    int256[] slippage,
    uint32 referralCode
    uint64 referralCode,
    uint64 referralFee,
    address referralFeeRecipient
);
```

The `registerReferralCode`, `referralLookup`, `referralNumManagers`, and `referralManager` functions have been deleted since all of them interacted with the deprecated ReferralRegistry contract.

5. Patch Review 2

This section documents notable changes applied to the in-scope code from commit d14d55fa to commit f90fd623 for our review on May 22nd to May 23rd, 2025.

5.1. Notable changes

The following notable changes were made to the codebase.

OdosWETHExecutor contract

The OdosWETHExecutor contract is used to handle WETH transfers. There has been no change to the contract.

OdosTransferHook contract

The OdosTransferHook contract has been added to the codebase. This contract is used to handle the transfer of tokens after a swap. While it has arbitrary token-transfer logic, all tokens are ultimately transferred to the destination address, so it is not possible to steal the tokens.

OdosRouterV3 contract

The `swapMultiPermit2WithHook`, `swapMultiWithHook`, and `swapWithHook` functions have been added to the OdosRouterV3 contract. These functions are similar to the existing swap functions but accept additional parameters: `hookTarget` and `hookData`. When the swap is executed and the tokens are transferred to the `hookTarget` contract, the OdosRouterV3 contract calls the `executeOdosHook` function to transfer its tokens to the address specified in `hookData`.

The referral fee now uses the `splitBPS` value instead of the fixed 80% fee. Additionally, positive slippage is now allowed when the 49th bit of the `referralCode` is set to 1.

```
uint256 splitBPS = (referralInfo.code >> 32) & 65535;
if (splitBPS == 0) splitBPS = 8000;
require(splitBPS <= 10000, "Invalid Ref Code");

  if (referralInfo.feeRecipient != address(this)) {
    _universalTransfer(
      tokenInfo.outputToken,
      referralInfo.feeRecipient,
      amountOut * referralInfo.fee * 8 / (FEE_DENOM * 10)
      amountOut * referralInfo.fee * splitBPS / (FEE_DENOM * 10000)
    );
  }
  amountOut = amountOut * (FEE_DENOM - referralInfo.fee) / FEE_DENOM;
```

```
    }
    int256 slippage = int256(amountOut) - int256(tokenInfo.outputQuote);
-   if (slippage > 0) {
+   if (slippage > 0 && (referralInfo.code >> 48) & 1 == 0) {
      amountOut = tokenInfo.outputQuote;
    }
```

6. Assessment Results

During our assessment on the scoped Odos Router V3 contracts, there were no security vulnerabilities discovered.

6.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.