















DeFi Liquidity Zapper

8/28/2025 • 1 files • 12 issues found

After a professional audit?

A Disclaimer: This Al-generated scan may contain inaccuracies and false positives. For professional audits, consult Hashlock security audit experts.

Contract Overview Description LiquidityZapETH is a smart contract that facilitates one-click liquidity provision to Uniswap V2 pools. It accepts ETH deposits, allocates 10% to a treasury address, and uses the remaining 90% to create liquidity positions in OEC/ETH pairs. The contract optimally splits the ETH between swapping for tokens and providing liquidity to minimize slippage and maximize LP token output. Access Control Ownable **Privileged Roles** owner **External Calls** IUniswapV2RouterO2 IUniswapV2Factory IUniswapV2Pair IERC20 (OEC Token) WETH External Systems **Uniswap V2 DEX Treasury Address**



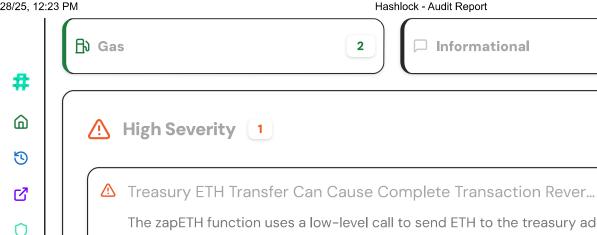






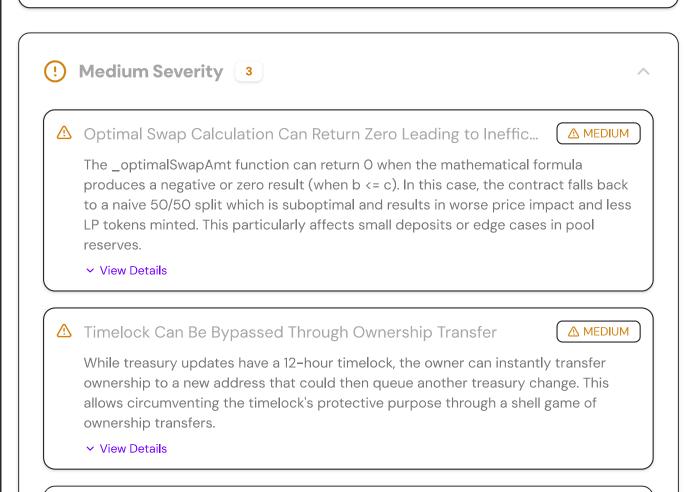


View Call Graph



The zapETH function uses a low-level call to send ETH to the treasury address without proper error handling. If the treasury is a contract that reverts on receive (e.g., due to gas limit, logic failure, or malicious behavior), the entire zap transaction will revert, causing users to lose gas and preventing them from providing liquidity. This creates a denial-of-service vector where a compromised or malfunctioning treasury can brick the entire protocol.

View Details



The contract calculates expected output and slippage protection based on reserves queried at the beginning of the transaction. However, these reserves can change between the query and actual swap execution due to other transactions in the same



Slippage Calculation Based on Stale Reserves

block, leading to unexpected slippage or failed transactions.

1

⚠ HIGH

→ View Details













i Low Severity 5

🛆 Incorrect Error Message on Treasury Transfer Failure

⚠ LOW

When the treasury ETH transfer fails, the contract reverts with 'BadRouterRefund()' error instead of a more appropriate error. This misleading error message could confuse users and developers debugging failed transactions, as the failure is related to treasury transfer, not router refunds.

View Details

▲ MIN_ETH_FOR_ZAP Too Low Allows Dust Griefing



The MIN_ETH_FOR_ZAP constant is set to only 2 wei, which is effectively no minimum. This allows griefing attacks where attackers can spam the contract with dust amounts that still emit events and consume gas but provide negligible liquidity. Additionally, such small amounts can cause precision loss in calculations and potentially revert due to Uniswap's minimum liquidity requirements.

View Details

A Receive Function Allows Unrestricted ETH From WETH Contract



The receive function accepts ETH from both the router and WETH contracts. However, WETH can be called by anyone to unwrap ETH to any address. An attacker could unwrap WETH to this contract, potentially interfering with accounting or causing unexpected behavior when the contract calculates leftover balances.

View Details



△ No Validation of Router Liquidity Return Values Beyond Minimu...



The contract checks that returned amounts meet minimum thresholds but doesn't validate that the router actually used all provided tokens/ETH efficiently. If the router has a bug or is upgraded maliciously, it could return valid but suboptimal amounts while keeping the difference.

View Details



No Mechanism to Recover Accidentally Sent Tokens



If users accidentally send ERC20 tokens directly to the contract (not through zapETH), there's no way to recover them. While the contract is designed to handle



#

 \bigcirc

(7)

ď

 \bigcirc

Hashlock - Audit Report

ETH and specific token interactions, it lacks a recovery mechanism for mistakenly sent tokens.

View Details

Gas Severity 2

☐ Informational Severity 1

Need a Professional Security Audit?

While Al provides quick insights, professional auditors catch what Al misses

- Hashlock Credentials
- Trusted by 1000's of protocols
- Auditing some of the world's leading protocols
- Comprehensive and formally recognised audit reports



















Manual code review by security experts

Detailed remediation guidance

Compliance-ready reports

Ongoing security partnership



Developed by ##

Built in Collaboration with