CODESPECT

Swell Withdrawal Update

SECURITY ASSESSMENT REPORT

14 April, 2025

Prepared for

Swell

Contents

1 About CODESPECT

CODESPECT is a specialized smart contract security firm dedicated to ensure the safety, reliability, and success of blockchain projects. Our services include comprehensive smart contract audits, secure design and architecture consultancy, and smart contract development across leading blockchain platforms such as Ethereum (Solidity), Starknet (Cairo), and Solana (Rust).

At CODESPECT, we are committed to build secure, resilient blockchain infrastructures. We provide strategic guidance and technical expertise, working closely with our partners from concept development through deployment. Our team consists of blockchain security experts and seasoned engineers who apply the latest auditing and security methodologies to help prevent exploits and vulnerabilities in your smart contracts.

Smart Contract Auditing: Security is at the core of everything we do at CODESPECT. Our auditors conduct thorough security assessments of smart contracts written in Solidity, Cairo, and Rust, ensuring that they function as intended without vulnerabilities. We specialize in providing tailored security solutions for projects on EVM-compatible chains and Starknet. Our audit process is highly collaborative, keeping clients involved every step of the way to ensure transparency and security. Our team is also dedicated to cutting-edge research, ensuring that we stay ahead of emerging threats.

Secure Design & Architecture Consultancy: At CODESPECT, we believe that secure development begins at the design phase. Our consultancy services offer deep insights into secure smart contract architecture and blockchain system design, helping you build robust, secure, and scalable decentralized applications. Whether you're working with Ethereum, Starknet, or other blockchain platforms, our team helps you navigate the complexity of blockchain development with confidence.

Tailored Cybersecurity Solutions: CODESPECT offers specialized cybersecurity solutions designed to minimize risks associated with traditional attack vectors, such as phishing, social engineering, and Web2 vulnerabilities. Our solutions are crafted to address the unique security needs of blockchain-based applications, reducing exposure to attacks and ensuring that all aspects of the system are fortified.

With a focus on the intersection of security and innovation, CODESPECT strives to be a trusted partner for blockchain projects at every stage of development and for each aspect of security.

2 Disclaimer

Limitations of this Audit: This report is based solely on the materials and documentation provided to CODESPECT for the specific purpose of conducting the security review outlined in the Summary of Audit and Files. The findings presented in this report may not be comprehensive and may not identify all possible vulnerabilities. CODESPECT provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, is entirely at your own risk.

Inherent Risks of Blockchain Technology: Blockchain technology is still evolving and is inherently subject to unknown risks and vulnerabilities. This review focuses exclusively on the smart contract code provided and does not cover the compiler layer, underlying programming language elements beyond the reviewed code, or any other potential security risks that may exist outside of the code itself.

Purpose and Reliance of this Report: This report should not be viewed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. Third parties should not rely on this report for any purpose, including making decisions related to investments or purchases.

Liability Disclaimer: To the maximum extent permitted by law, CODESPECT disclaims all liability for the contents of this report and any related services or products that arise from your use of it. This includes but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services: CODESPECT does not warrant, endorse, or assume responsibility for any third-party products or services mentioned in this report, including any open-source or third-party software, code, libraries, materials, or information that may be linked to, referenced by, or accessible through this report. CODESPECT is not responsible for monitoring any transactions between you and third-party providers. We strongly recommend conducting thorough due diligence and exercising caution when engaging with third-party products or services, just as you would for any other product or service transaction.

Further Recommendations: We advise clients to schedule a re-audit after any significant changes to the codebase to ensure ongoing security and reduce the risk of newly introduced vulnerabilities. Additionally, we recommend implementing a bug bounty program to incentivize external developers and security researchers to identify and disclose potential vulnerabilities safely and responsibly.

Disclaimer of Advice: FOR AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, AND ANY ASSOCIATED SERVICES OR MATERIALS SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Risk Classification

Severity Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Table 1: Risk Classification Matrix based on Likelihood and Impact

3.1 Impact

- **High** - Results in a substantial loss of assets (more than 10%) within the protocol or causes significant disruption to the majority of users.
- **Medium** - Losses affect less than 10% globally or impact only a portion of users, but are still considered unacceptable.
- **Low** - Losses may be inconvenient but are manageable, typically involving issues like griefing attacks that can be easily resolved or minor inefficiencies such as gas costs.

3.2 Likelihood

- **High** - Very likely to occur, either easy to exploit or difficult but highly incentivized.
- **Medium** - Likely only under certain conditions or moderately incentivized.
- **Low** - Unlikely unless specific conditions are met, or there is little-to-no incentive for exploitation.

3.3 Action Required for Severity Levels

- **Critical** - Must be addressed immediately if already deployed.
- **High** - Must be resolved before deployment (or urgently if already deployed).
- **Medium** - It is recommended to fix.
- **Low** - Can be fixed if desired but is not crucial.

In addition to High, Medium, and Low severity levels, CODESPECT utilizes two other categories for findings: **Informational** and **Best Practices**.

a) **Informational** findings do not pose a direct security risk but provide useful information the audit team wants to communicate formally.

b) **Best Practices** findings indicate that certain portions of the code deviate from established smart contract development standards.

4 Executive Summary

This document presents the security assessment conducted by CODESPECT for the smart contracts of Swell. Swell is a non-custodial staking protocol offering liquid staking and restaking tokens.

This audit focuses on a minor update to two contracts related to EigenLayer integration. The update was necessitated by upcoming changes in the EigenLayer codebase. These modifications involve the support of changes in EigenLayer interface.

The audit was performed using:

a) Manual analysis of the codebase.

CODESPECT did not identify any security concerns during the audit. A summary of the changes related to the Swell and EigenLayer codebase can be found in **Section 6**.

Organization of the document is as follows:

- **Section 5** summarizes the audit.
- **Section 6** describes the update of EigenLayer and Swell code.
- **Section 7** discusses the documentation provided by the client for this audit.
- **Section 8** presents the compilation and tests.

Issues found:

Severity	Unresolved	Fixed	Acknowledged
Total	0	0	0

Table 2: Summary of Unresolved, Fixed, and Acknowledged Issues

5 Audit Summary

Audit Type	Security Review
Project Name	Swell
Type of Project	EigenLayer Integration Update
Duration of Engagement	2 Days
Duration of Fix Review Phase	Not applicable
Draft Report	April 12, 2025
Final Report	April 14, 2025
Repository	v3-contracts-lrt
Commit (Audit)	78b0154d227490e76c02a89607449acdb0138fa6
Commit (Final)	78b0154d227490e76c02a89607449acdb0138fa6
Documentation Assessment	High
Test Suite Assessment	Not Applicable
Auditors	Talfao, JecikPo

Table 3: Summary of the Audit

5.1 Scope - Audited Files

	Contract	LoC
1	EigenLayerManager.sol	446
2	StakerProxy.sol	227
	Total	**673**

Scope information

Only the code changes outlined in PR-106 were considered within the scope of this audit. Specifically, the assessment focused solely on the relationship between the scoped files and the EigenLayer update.

5.2 Findings Overview

CODESPECT did not identify any issues in the reviewed codebase.

6 Swell Update Information

Swell introduced new changes to its `EigenLayerManager` and `StakerProxy` contracts due to the Eigen update to M4 contracts. M4 contracts introduce slashing, which is intended to penalize operator shares as a protection mechanism against malicious behaviour that could compromise economic security.

Swell introduced only a few changes, as the in-scope items are not significantly affected by this update due to the centralization of withdrawals and the pricing model of the `rswETH` token. These components depend on input from privileged accounts; therefore, only the correctness of those inputs needs to be ensured. **Swell is expected to handle all changes, particularly those related to potential slashing.**

Swell's changes incorporate modifications to the Delegation and Withdrawal interfaces.

6.1 Withdrawal Changes

The `completeWithdrawal(...)` function, which is called on `DelegationManager`, no longer includes the `uint256 _middlewareTimesIndex` parameter. Swell has addressed this correctly in its `StakerProxy` implementation:

```
function completeQueuedWithdrawal(
  IDelegationManager.Withdrawal calldata _withdrawal,
  IERC20[] calldata _tokens,
  bool _receiveAsTokens
) external checkEigenLayerManager(msg.sender) {
  DelegationManager.completeQueuedWithdrawal(
    _withdrawal,
    _tokens,
    _receiveAsTokens
  );
}
```

Swell imports the updated interface, which reflects changes in the `Withdrawal` and `QueuedWithdrawalParams` structs:

```
struct QueuedWithdrawalParams {
    IStrategy[] strategies;
    uint256[] depositShares;
    address __deprecated_withdrawer;
}

struct Withdrawal {
    address staker;
    address delegatedTo;
    address withdrawer;
    uint256 nonce;
    uint32 startBlock;
    IStrategy[] strategies;
    uint256[] scaledShares;
}
```

The original `withdrawer` field in `QueuedWithdrawalParams` was renamed to `__deprecated_withdrawer`, and this attribute is no longer used in EigenLayer's `queueWithdrawals(...)` function. The `Withdrawal` struct replaced `shares` with `scaledShares`, which should reflect the `depositedShares`. These values do not account for slashing directly; the actual returned assets may differ due to slashing events which happened in prior and after queuing the withdrawal.

Administrators must handle slashing values off-chain to ensure correct pricing of the `rswETH` token.

6.2 Delegation

The main interface change in delegation is the deprecation of the previous method used to delegate a staker to an operator using the staker's signature. Now, only the `delegateTo(...)` function is used:

```
function delegateToOperator(
  address _operator,
  IDelegationManager.SignatureWithExpiry calldata _approverSignatureAndExpiry,
  bytes32 _approverSalt
) external checkEigenLayerManager(msg.sender) {
  DelegationManager.delegateTo(
    _operator,
    _approverSignatureAndExpiry,
    _approverSalt
  );
}
```

This change removes the staker's signature component, retaining only parameters related to the operator and approver's signature.

The current interface also allows undelegation and re-delegation, which remain unchanged. However, Swell's codebase does not contain a function to re-delegate the operator.

6.3 Security and Update Considerations

EigenLayer highlighted several important considerations in its update proposals that Swell must take into account:

- *Stakers will see changes to their risk and reward profile following this upgrade. Initially following the upgrade, if a Staker is already delegated to an Operator, its stake can become slashable as soon as the Operator opts in to Operator Sets and allocates stake. This will create risk (and potential return). Stakers therefore should review and confirm their risk tolerances for their continued delegations to operators.*
  - Swell must evaluate whether its operator will allocate stake and if participation in this risk is acceptable.
- *There exists a case where EigenPod contracts become fully inoperable when a checkpoint is completed after all attached validators are fully slashed on the beacon chain (100%) and there are no other assets on the EigenPod. It will become "bricked" or non-functional, meaning even if new assets are added or validators attached to the Pod, it will still remain non-functional.*
  - Swell should be aware of this risk and avoid depositing into fully slashed pods. However, as stated by EigenLayer, the likelihood of this scenario is very low.
- *EigenPod stakers should primarily be aware of the temporal pause in checkpoints [ref].*
  - Swell should be aware of this temporary protocol behavior.
- Swell has to handle its pricing of `rswETH` token correctly. This pricing has to reflect all of the slashing events which affect stakers' deposits.

6.4 References

- ELIP-002
- ELIP-004

7 Evaluation of Provided Documentation

The Swell documentation was provided in the form of the PR-106 description, which clearly outlined the changes made to the codebase. Throughout the evaluation process, the Swell team remained consistently available and responsive, promptly addressing all questions raised by CODESPECT.

8 Test Suite Evaluation

Not applicable in this case, as the evaluation of the test suite was not performed due to the nature, duration, and scope of the audit.