Code Assessment
of the Grove User Actions Smart Contracts

PUBLIC

Date May 05, 2026

Produced for

By

Contents

1 Executive Summary

Dear all,

Thank you for trusting us to help GroveLabs with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Grove User Actions according to **Scope** to support you in forming an opinion on their security risks.

GroveLabs implements a contract that batches actions of the PSM and the savings token. This is a fork of **Spark User Action**.

The most critical subjects covered in our audit are functional correctness and precision of arithmetic operations.

Security regarding all the aforementioned subjects is high.

The general subjects covered are unit testing and documentation.

There is no documentation in the codebase.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,
ChainSecurity

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Grove User Actions repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

Version	Date	Commit Hash	Note
1	24 April 2026	6fb59651b26d6d36d5c25d0dbc50a5e88f62463e	Initial Review

For the Solidity smart contracts, the compiler version `0.8.25` was chosen.

```
src/
└── PSMVariant1Actions.sol
```

The code was compared against sparkdotfi/spark-user-actions at commit `ca51ba8f0a5a7bb052c0b45a525abd6b38c976a8` (latest reviewed version at the time of writing this report).

2.1.1 Excluded from scope

Any files not listed above are out of scope.

3 System Overview

This system overview describes the initially received version (**Version 1**) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

GroveLabs offers peripheral contracts that act as wrappers for batching certain functionalities.

3.1 PSMVariant1Actions

The `PSMVariant1Actions` batches PSM swaps (USDS PSM Wrapper) and ERC-4626 deposits/withdrawals (e.g. sUSDS).

The `PSMVariant1Actions` contract implements the following function:

1. `swapAndDeposit`: Pulls the PSM's gem token from the caller and sells it to the PSM for its `dai` token (USDS) which it it then deposits to the ERC-4626 savings token with a referal code `2001`.
2. `withdrawAndSwap`: Withdraws an amount of the `dai` (USDS) from the caller's savings so that it can be swapped for an exact amount of the gem.
3. `redeemAndSwap`: Redeems an exact amount of the savings token and swaps the received underlying token to the gem. Note that this operation might leave some dust in the contract.

All operations implement slippage protection in case the PSM's `tout()` and, thus, the fees change. The slippage protection for `swapAndDeposit` does not consider fees on deposit. Note that these are not expected (e.g. sUSDS has no fees).

4 Trust Model

All the interacted with addresses (e.g. `psm`, `dai`, `gem`, `savingsToken`) are expected to be the correct addresses and are trusted and expected to work correctly. Users are untrusted.

There are no privileged roles defined for the contracts.

5 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- **Likelihood** represents the likelihood of a finding to be triggered or exploited in practice
- **Impact** specifies the technical and business-related consequences of a finding
- **Severity** is derived based on the likelihood and the impact

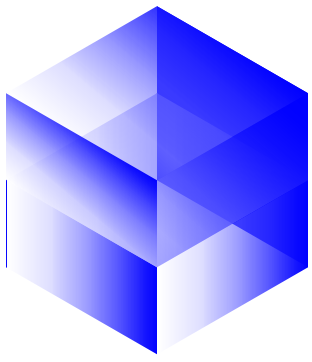
We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	**Impact**		
	High	Medium	Low
High	**Critical**	**High**	**Medium**
Medium	**High**	**Medium**	**Low**
Low	**Medium**	**Low**	**Low**

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

6 Findings

In this section, we describe the findings identified during the course of the engagement. The findings are categorized by severity as explained in the Terminology section. Below we provide an overview of the total number of findings per severity, as well as the number of open issues.



No Open Issues

Total number of findings

Critical	High	Medium	Low	Informational
0	0	0	0	0

7 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.