0xCommit

Security Assessment Report

Perpie

Perpie V2 Security Assessment Report

Version: Draft

Date: 8 Feb 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

Introduction

Purpose of this report

0xCommit has been engaged by **Perpie** to perform a security audit of several Smart Account components.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behaviour.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebases Submitted for the Audit

The audit has been performed on the following commits:

Github Link:

Version	Commit hash
Initial	
Final	

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
High	An attacker can successfully execute an attack that clearly results in operational issues for the service. This also includes any value loss of unclaimed funds permanently or temporary.
Medium	The service may be susceptible to an attacker carrying out an unintentional action, which could potentially disrupt its operation. Nonetheless, certain limitations exist that make it difficult for the attack to be successful.
Low	The service may be vulnerable to an attacker executing an unintended action, but the impact of the action is negligible or the likelihood of the attack succeeding is very low and there is no loss of value.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary

The status of an issue can be one of the following: **Pending**, **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Overview

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
    d. Access Control Issues
    e. Boundary Analysis
4. Report preparation

Functionality Overview

Summary of Findings

Sr. No.	Description	Severity	Status
1	Zero Prices in usdtotoken Function of PriceFeed Contract	Low	Resolved
2	Initializing Ownable Contract in GMX FeesModule Constructor	Informatio...	Resolved
3	Evaluating the Upgradability of FeeManager Contract	Medium	Pending
4	Optimizing PriceFeed Contract	Informatio...	Resolved
5	Token transfer return value not factored in execute function in GMX fees module	Informatio...	Resolved

Detailed Findings

1. Zero Prices in **usdtotoken** Function of PriceFeed Contract

Severity: Low

Description

PriceFeed contract, the function **usdtotoken** serves a crucial role in converting USD to tokens. However, finding highlights a vulnerability: the possibility of encountering a zero price. This vulnerability could lead to significant issues such as incorrect token valuations or even potential exploitation by malicious actors.

Remediation

Update the **usdtotoken** function to include robust checks for zero prices, implementing fail-safe mechanisms and alternative pricing strategies.

Status

Pending

2. Initializing Ownable Contract in **GMX FeesModule** Constructor

Severity: Informational

Description

A potential vulnerability within the GMX FeesModule contract, wherein the Ownable contract is not initialized in the constructor. By not initializing the Ownable contract in the constructor, there's a risk of leaving the contract vulnerable to unauthorized access or manipulation, as ownership control mechanisms may not be properly established.

Remediation

Ensure proper initialization of the Ownable contract within the constructor of the GMX FeesModule contract to establish ownership control mechanisms.

Status

Pending

3. Evaluating the Upgradability of **FeeManager** Contract

Severity: Medium

Description

Check if the FeeManager can be upgraded or not. If it can be upgraded, we should see if developers are actually using this feature. If they aren't, leaving it upgradable could cause potential problems, which are Unauthorized upgrades, vulnerabilities in dependencies, changes in governance, erosion of trust, and emerging threats could compromise the contract's security.

Remediation

If developers anticipate needing this feature in the future, ongoing monitoring and proactive measures are necessary to mitigate these risks and ensure the contract remains secure and reliable.

If developers are not using this feature, it is recommended that the contract should be updated and rendered non-upgradable.

Status

Pending

4. Optimizing **PriceFeed** Contract

Severity: Informational

Description

Redundant or repetitive code related to timestamp can cause inefficiencies in gas usage, potentially leading to higher transaction costs and slower contract execution times.

Remediation

Optimizing **PriceFeed** contract by maximizing efficiency through the reuse of timestamp functions can be a viable fix to this problem.

Status

Pending

5. Token transfer return value not factored in execute function in GMX fees module

Severity: Informational

Description

By not factoring in the return value of token transfers, the execute function may fail to accurately determine the outcome of transactions. This oversight could lead to discrepancies in fee calculations.

Remediation

Modify the execute function to properly handle the return value of token transfers, ensuring accurate transaction outcomes and fee calculations. Implement error handling mechanisms to detect and address any failures or discrepancies in token transfers.

Status

Pending