0xCommit

Security Assessment Report

Kwenta

Version: Final

Date: 25 Oct 2023

Table of Contents

Licence

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 **Kwenta** to perform a security audit of several Account Abstraction Session Validation 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 GitHub repositories:

Repository : https://github.com/Kwenta/scw-contracts

Contract in scope :

- ➔ SMv2SessionValidationModule.sol
- ➔ SMv3SessionValidationModule.sol

Version	Commit hash
Initial Codebase	d0ce77bb16eca30aa6fe4a155f7ecc60469f95c3
Fixed Codebase	da8c9449ff60b4fad13b2dcd4935c6eeb01cdffb

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

The Session Validation Module plays a critical role in verifying the specifics of a User Operation, which delineates an interaction between a user and either Kwenta's Smart Margin v2 or Smart Margin v3 system. To accommodate Account Abstraction, Kwenta has implemented a customized version of the Session Validation Module.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Flexible Function Selector Inclusion in SMv3SessionValidationModule	Informational	Resolved

Detailed Findings

1. Flexible Function Selector Inclusion in the validation modules

Severity: Informational

Description

In SMv3SessionValidationModule, the current implementation assesses whether the function selector in the calldata corresponds to one of four established selectors:

- *smv3ModifyCollateralSelector*
- *smv3CommitOrderSelector*
- *smv3InvalidateUnorderedNoncesSelector*
- *smv3FulfillOracleQuerySelector*

There is a similar case in SMv2SessionValidationModule, where it expects the calldata function selector to be *smv2ExecuteSelector*.

It's important to note that these selectors are introduced into the session data by the smart account owner, who retains the ability to introduce alternative selectors beyond these four.

While this does not present a critical concern, it does underscore the importance of considering the implications of this flexible selector inclusion to ensure the system's security and integrity.

Remediation

To address this informational-level issue in the SMv3SessionValidationModule, we recommend implementing one of the following measures:

➔ **Restrictions on Session Key Data**: Introduce restrictions or validation checks when adding session key data to limit the addition of arbitrary function selectors.

➔ **Calldata vs. Interface Function Selector Comparison**: An alternative approach is to compare the function selectors found in the calldata with the actual function selectors defined in the interface. This validation step ensures that the selectors used in the session data correspond to the expected and authorized function selectors.

Status

Resolved

The Kwenta team decided to follow the suggestion and compare the function selectors in the calldata with those defined in the interface.