CANTINA

Maker Dao - Spark Conduits

Security Review

Cantina Managed review by:
Christoph Michel, Lead Security Researcher
M4rio.eth, Security Researcher

November 24, 2023

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

The Maker Protocol, also known as the Multi-Collateral Dai (MCD) system, allows users to generate Dai (a decentralized, unbiased, collateral-backed cryptocurrency soft-pegged to the US Dollar) by leveraging collateral assets approved by the Maker Governance, which is the community organized and operated process of managing the various aspects of the Maker Protocol.

From Oct 10th to Oct 16th the Cantina team conducted a review of sparklend-conduits on commit hash fdb54d1b.

Important note

Context: Updated Contracts

Description: During the audit, the client has decided to remove most of the variable interest functionality. The client has stated that they plan to list only sNST in the SparkLend, which will be using a `DefaultReserveInterestRateStrategy` (an example of this strategy can be seen currently used for sDAI here) with a flat curve to be used as the debt is denominated in the savings version which accounts for the DSR.

The decision behind the change was made mostly to the various imperfections and edge-casing that made the system very prone to unwanted scenarios.

Additionally, the team has stated that they do not expect for the interest rate to be modified anytime soon, this could happen only in Black Swan events, if needed a new version will be deployed accordingly that it will go through the governance vote.

***Additional note**: The original review commit hash was 2d559911, but the final commits that were signed off correspond to commit hash fdb54d1b as stated above.*

The team identified a total of **5** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 0
- Informational: 5

3 Findings

3.1 Informational

3.1.1 Invariants are not strict equalities because of different rounding between Conduit and Spark

Severity: Informational

Context: README.md#invariants

Description: The invariant section defines the following invariants with this note:

$$totalShares[asset] \leq aToken.scaledBalanceOf(conduit)$$

$$getTotalDeposits(asset) \leq aToken.balanceOf(conduit)$$

> **NOTE**: The last two invariants are not strict equalities because of the potential for a permissionless transfer of the aToken into the conduit. For this reason alone, they are expressed as inequalities.

The reason these are inequalities is not alone because of permissionless transfers to the conduit. These are inequalities even if no direct aToken transfer has been performed because of a difference in rounding: Spark rounds half-up (up if > 0.5). For example, when `deposit`ing the shares, Conduit computes:

```
uint256 newShares = _convertToShares(asset, amount) = "amount.rayDivDown(liquidityIndex)"
```

whereas Spark rounds half-up in `mintScaled` (which is called when supplying `amount`):

```
uint256 amountScaled = amount.rayDiv(index) = "amount.rayDivRound(index)"
```

See WadRayMath.rayDiv.

It can happen that the `shares` are rounded down in `SparkConduit` but rounded up in Spark and therefore $\text{totalShares[asset]} \leq \text{aToken.scaledBalanceOf(conduit)}$. However, we agree with the rounding direction of `SparkConduit`, always choosing the conservative amounts (rounding against the user, s.t., there are always more shares on Spark than on `SparkConduit`).

Recommendation: Consider correcting the note about why the invariants are inequalities.

Maker: Implemented the recommended change in commit 9ff06d1d.

Cantina: Fixed.

3.1.2 Custom division function `_divup` misbehaves when `0` is divided by `0`

Severity: Informational

Context: SparkConduit.sol#L185

Description: `_convertToSharesRoundUp` returns `0` when `0` is divided by `0`. The expected behaviour aligned with the Solidity division operator should be a revert.

```
function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
    unchecked {
        z = x != 0 ? ((x - 1) / y) + 1 : 0;
    }
}
```

As seen above, `x == 0` will return `0`, even if y is also a `0`. However, this function is only called as `shares = _divUp(amount * 1e27, IPool(pool).getReserveNormalizedIncome(asset))` and the normalized income is never `0`. So there is no direct impact.

Recommendation: This function is used in many Maker code bases. Consider fixing or documenting this feature so there is no bug introduced if this function is used elsewhere.

Maker: Implemented the recommended change in commit 19bb5543.

Cantina: Fixed.

3.1.3 No inherent reward handling

Severity: Informational

Context: SparkConduit.sol#L110

Description: Spark supports rewards as an Aave-V3 fork. The contract currently has no inherent way to claim the rewards and it isn't easy to automatically distribute them among the subDAOs *fairly* (taking into account supplied time and not just the supplied amount).

Recommendation: One could consider adding an `auth`'d function that calls `claimRewards()` to a recipient that correctly distributes them according to off-chain calculations. Alternatively, consider using the Spark-controlled `EmissionManager` to set a claimer for this contract to a trusted party once rewards for this contract become an issue. This has the upside of this contract not requiring any reward-claiming logic itself.

Maker: If this ever becomes a valid use case we will upgrade the contract.

Cantina: Acknowledged.

3.1.4 No donations supported

Severity: Informational

Context: SparkConduit.sol#L176

Description: The shares' value is determined by an external price (Spark's liquidity index, `IPool(pool).getReserveNormalizedIncome(asset)`). This means the share price can *not* be increased by directly transferring underlying to the contract, as one might expect from similar vault-like contracts.

Recommendation: Users and integrators should be aware that funds allocated to this conduit cannot be increased through donations and directly transferred funds will be stuck.

Maker: We don't expect this to happen and if it does at any meaningful size we can address it with an implementation upgrade.

Cantina: Acknowledged.

3.1.5 Unused `PotLike` interface

Severity: Informational

Context: SparkConduit.sol#L13-L15

Description: The `PotLike` interface is unused within the current implementation.

Recommendation: Consider removing this interface.

Maker: Implemented in commit 16ab9073.

Cantina: Fixed.

4 Additional Comments

The Cantina team reviewed MakerDao's sparklend-conduits changes holistically on commit hash 2d559911963ca6e5fde88c46ff22ec7b2e515ead and determined that all issues were resolved and no new issues were identified.