Fixed Point Solutions, LLC

dss-allocator Assessment

2023/08/30
Prepared by: Kurt Barry

1. Scope

Smart contracts in the src/ directory of pull request 41 of the makerdao/dss-allocator repo were reviewed for safety and correctness. The review began at commit 36e0830379b1c8b54630abb79c5077d4e9dad200 and covered all subsequent commits in the PR until commit a5469884813528b414ec1c2b985f52dd192455a1. The review was conducted by a single security researcher using approximately 1.5 days of total effort from August 11th, 2023 until August 30th, 2023.

Uniswap V3 integration was not covered in detail; rather, it was assumed that Uniswap interfaces worked "as assumed" by the code. The files FullMath.sol, LiquidityAmount.sol, and TickMath.sol were checked by diffing them against their original versions and ensuring the changes made did not alter their functionality; the mathematical algorithm implementations were not themselves assessed.

2. Limitations

No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3. Findings

Findings and recommendations are listed in this section, grouped into broad categories. It is up to the team behind the code to ultimately decide whether the items listed here qualify as issues that need to be fixed, and whether any suggested changes are worth adopting. When a response from the team regarding a finding is available, it is provided.

Findings are given a severity rating based on their likelihood of causing harm in practice and the potential magnitude of their negative impact. Severity is only a rough guideline as to the risk an issue presents, and all issues should be carefully evaluated.

Severity Level Determination		Impact		
		High	Medium	Low
Likelihood	High	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**

Issues that do not present any quantifiable risk (as is common for issues in the Code Quality category) are given a severity of **Informational**.

3.1 Safety and Correctness

Findings that could lead to harmful outcomes or violate the intentions of the system.

SC.1 Incorrect Check In DepositorUniV3 Could Block Deposits

Severity: Low

Code Location:
https://github.com/makerdao/dss-allocator/blob/aea59a10983470793f315638be94cfc66f913b4d/src/funnels/DepositorUniV3.sol#L200

Description: The indicated line checks whether the amount of token zero owed is non-zero in order to decide whether to perform a transfer of token one; in a case where only token one is owed, the transfer would not happen as intended, causing the Uniswap mint function to revert, preventing the deposit.

Recommendation: Check the amount of token one on this line instead.

Response: Fixed in commit 40005875d2eb3f6dd67b6267e6091d75e0eed565.

SC.2 Tokens With Special Behaviors Are Not Supported

Severity: Low

Code Location: Token transfer logic used throughout the codebase.

Description: Various unusual token behaviors (for example, returning false for a failed transfer instead of reverting, assessing a fee on transfers, or rebasing) may result in bugs or compatibility issues.

Recommendation: Document what sorts of tokens are intended to be compatible with these contracts, and implement logic to support non-standard token behaviors if needed.

Response: Tokens with special behaviors will not be used with these contracts; intent documented in commit a3d2095a484f74b17b0edd6ebc0afd6ffc7862a7.

3.2 Usability and Incentives

Findings that could lead to suboptimal user experience, hinder integrations, or lead to undesirable behavioral outcomes.

U.1 Consider Whether a hasActionRole Function Would Be Useful In AllocatorRoles

Severity: Informational

Code Location:
https://github.com/makerdao/dss-allocator/blob/7923bcc2254566886da50c2e2a29a7251c44e2e0/src/AllocatorRoles.sol#L58

Description: The AllocatorRoles contract contains a convenience method for checking whether a user is assigned to a particular role. However, no analogous method for checking whether an action is assigned to a role exists.

Recommendation: Consider adding a method to check whether and action is assigned a role.

Response: Added such a method in commit a24fc1387e8ce24d0f695f6714f67cb3e9bc29c2.

3.3 Gas Optimizations

Findings that could reduce the gas costs of interacting with the protocol, potentially on an amortized or averaged basis.

G.1 Checked Arithmetic in TickMath.sol May Be Unnecessary

Severity: **Informational**

Code Location:
[1]https://github.com/makerdao/dss-allocator/blob/40005875d2eb3f6dd67b6267e6091d75e0eed565/src/funnels/uniV3/TickMath.sol#L28
[2]https://github.com/makerdao/dss-allocator/blob/40005875d2eb3f6dd67b6267e6091d75e0eed565/src/funnels/uniV3/TickMath.sol#L97

Description: In the original Uniswap code, the `getSqrtRatioAtTick` and `getTickAtSqrtRatio` functions used unchecked arithmetic. The later compiler version used here (0.8.16) means that now these functions are using checked arithmetic. Assuming the original code was correct as-written, this should be unnecessary and could even cause unintentional reverts if at any point overflow or underflow is intended (no such operations were identified).

Recommendation: If additional gas efficiency is desired, consider wrapping the bodies of these functions in `unchecked` blocks.

Response: Changed to unchecked arithmetic in commit a5469884813528b414ec1c2b985f52dd192455a1.

3.4 Code Quality

CQ.1 Division Is Less Readable Than Explicit Right Shifting For Bit Manipulation

Severity: **Informational**

Code Location:
https://github.com/makerdao/dss-allocator/blob/aea59a10983470793f315638be94cfc66f913b4d/src/funnels/callees/SwapperCalleeUniV3.sol#L51-L52

Description: These lines divide by 2^{96} to right-shift the bits within calldata values. While there is nothing incorrect about this, an explicit right shift operator (`shr`) exists in Yul and is more self-documenting than dividing by a “magic” number. In particular, visually counting zeros is difficult for humans, although the values could be expressed in a more obvious fashion; however, using division is still less clear when it comes to communicating intent to readers of the code.

Recommendation: Consider using explicit right (or left) shifts when doing bit manipulation as appropriate.

Response: Obsolete since the logic has been removed in commit e8275c8c435f5fdc012e8f0a52e8004b3c4c3b3a.

4. Notes

This section contains general considerations for interacting with or maintaining the system and various conclusions reached or discoveries made during the course of the assessment. Whereas findings generally represent things for the team to consider changing, notes are more informational and may be helpful to those who intend to interact with the system.

4.1 Reentrancy in Swapper Contract

The Swapper contract's `swap()` function allows reentrancy. This behavior was examined in detail for problematic consequences, but nothing rising to the level of an attack or safety violation was identified. It is worth noting that if the `era` parameter for a token pair is zero, then the `cap` parameter is essentially meaningless as unlimited swaps can be done in a single transaction (until the block gas limit is exhausted, of course). The authors of the code have confirmed that this is intended behavior. It can also be pointed out that nested swaps may fulfill token transfer requirements in unusual ways (for example, the first of two swaps acquiring enough tokens to cover the outflow requirements for both, with the second not executing anything resembling a "swap" in the technical sense). This is only problematic if the overall value exchange is unfavorable, and if it is, reentrancy is not really the cause given the access-controlled nature of the Swapper (either parameters were configured poorly or a malicious actor was improperly trusted).