yAudit Vfat Sickle Review

Review Resources:

- None beyond the code repositories

Auditors:

- spalen
- Jackson

Table of Contents

Review Summary

Sickle

Sickle provides an extensible on-chain account. This on-chain account uses protocol-provided strategies to perform various on-chain functions, such as farming, sweeping tokens, re-balancing positions, or taking flash loans. The strategies themselves use shared modules and connectors to interact with defi protocols. This audit focused exclusively on a new rebalance strategy.

The contracts of the Sickle Repo were reviewed over 5 days. The code review was performed by 2 auditors between June 17, 2024 and June 21, 2024. The repository was under active development during the review, but the review was limited to the latest commit at the start of the review. This was commit 8e75747cd7d9520829abe615017f9914bbca5087 of the Sickle repo.

Note that the auditors originally reviewed 68498eb15f007a4d883084b1c1c3edc5aac86de9. However, a change was introduced such that a new `delayMin` variable in the `RebalanceConfig` was introduced. The auditors deemed this change small enough that the commit hash could be updated to 8e75747cd7d9520829abe615017f9914bbca5087.

Additional changes outside the scope of the initial review were reviewed via GitHub two PRs, 262, which covered changes from the initial scope and 263 which added a new `Automation.sol` contract.

A publicly accessible version of the repo with the fixes from this audit can be found here at commit 3dfdf44d3e105a90045226fea1b548f72e7960cd.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
.
├── RebalanceRegistry.sol
├── interfaces
│   └──IRebalanceRegistry.sol
├── libraries
│   └── RebalanceLib.sol
└── strategies
   └── RebalanceStrategy.sol
```

After the findings were presented to the Sickle team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, Sickle and users of the contracts agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	One of the challenges in attacking the Sickle protocol is that the Sickles are partitioned by user, and only whitelisted strategies can multi-call into the Sickles. These whitelisted strategies can only interact with whitelisted connectors.
Mathematics	Good	There are very few complex mathematical operations in the code reviewed.

Category	Mark	Description
Complexity	Medium	There is complexity around the flow of funds between the user, their Sickle, the flashloan provider, and the DeFi protocol that the Sickle interacts with.
Libraries	Average	Well-known Uniswap libraries are used.
Decentralization	Good	A user is free to interact with their Sickle and provided strategies as they see fit.
Code stability	Good	Changes were made to the code base during the review but not to the contracts in scope.
Documentation	Low	No documentation was provided for the in-scope contracts.
Monitoring	Average	The two state-changing functions contain events, however, `rebalance()` and `rebalanceFor()` do not.
Testing and verification	Average	The contracts in scope range from 84% to 100% test coverage.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
  - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
  - Findings that can improve the gas efficiency of the contracts.
- Informational
  - Findings including recommendations and best practices.

Critical Findings

None.

High Findings

None.

Medium Findings

None.

Low Findings

None.

Gas Savings Findings

None.

Informational Findings

1. Informational - Unused import

There is an unused import that can be safely removed.

Technical Details

INonfungiblePositionManager import in RebalanceRegistry.sol is unused.

Impact

Informational.

Recommendation

Remove unused imports.

Developer Response

Fixed in https://github.com/vfat-io/sickle-contracts/commit/48fcf31661cd2221fbdc6fbfb286b1ca9ff7d489.

2. Informational - Require rebalance config

Rebalance configuration can be unset and left with default values which will lead to reverts.

Technical Details

`RebalanceConfig` can be set and unset. If the configuration is left unset, function `rebalanceFor()` will revert with errors `TickWithinRange` or `TickOutsideMaxRange`.

Impact

Informational.

Recommendation

Revert if the configuration is not set to provide more descriptive error.

Developer Response

Fixed in https://github.com/vfat-io/sickle-contracts/commit/7d1762c1e8262bedd3e7ab4549629e7f1eaa0df4.

3. Informational - Require to use the whole position in `rebalanceFor()`

The function `rebalanceFor()` enables a user to automate rebalancing his Uniswap V3 position. If the whole amount is not used, the old position will be left and need user interaction to get it out and use it again.

Technical Details

The rebalance process consists of withdrawing the old position from the Uniswap V3 pool and minting a new position after which, the old rebalance configuration is deleted. If the whole old position is not withdrawn, it will be left outside the automation process and would need user interaction to withdraw it and use it.

Impact

Informational.

Recommendation

In function `rebalanceFor()` require that the whole position is withdrawn from Uniswap V3 position. This will ensure the user won't get fragmented by mistake and left outside the automation process.

Developer Response

Fixed in https://github.com/vfat-io/sickle-contracts/commit/c268a1848c65ee0aabfd1ee364e10a16f1a04931.

4. Informational - Harvest fees can be bypassed

Users can pass an empty array as param `harvestParams.tokensOut` to bypass paying harvest fees to the protocol.

Technical Details

Function `rebalance()` and `rebalanceFor()` charge harvest fees. This fee is charged using `FeesLib.chargeFees()` which has a param `feeTokens` to define array of tokens in which the fees are paid. If this array is empty, charging fees will be skipped. `feeTokens` are defined in the struct `HarvestParams` as variable `tokensOut`. If the user defines this param as an empty array, he will bypass paying any harvest fees.

Additionally, in the function `rebalance()`, param `pool` is used for fetching the pool fee. If the user is rebalancing the position in the pool with a high fee, he can set the `pool` param to another pool with a lower fee and pay lower fees to the protocol.

All valuable data is calculated off-chain in bulk with these parameters so changing only part of the parameters will require some effort. The function `rebalanceFor()` will be triggered by the bot, which will limit user changes. The user-triggered function `rebalance()` can be sniffed for valuable data from off-chain calculations. The user can change only parameters related to fees enabling him to pay less fees to the protocol.

Impact

Informational.

Recommendation

Rebalance fees, as well as other fees, can be bypassed, making it a system design. Be aware of this potential problem and enforce paying fees if needed.

Developer Response

Acknowledged as in the previous audit.

5. Informational - Use the same numeral system

For better readability, use the numeral system.

Technical Details

Tick value is loaded using assembly code. It moves the pointer two times to get the correct value but uses two different numeral systems for the same value.

Impact

Informational.

Recommendation

Use the same numeral system:

```
tick := mload(add(add(result, 32), 32))
```

Developer Response

Fixed in https://github.com/vfat-io/sickle-contracts/commit/8daae30eaf618e7f305a57f6ceac277f20d4742b.

6. Informational - Missing event emits

Some functions that transfer values do not have events. Events can assist with analyzing the on-chain history of contracts and are therefore beneficial to add in important functions.

Technical Details

Functions that could have events added include:

- `rebalance()`
- `rebalanceFor()`

Impact

Informational.

Recommendation

Add events to the functions listed above.

Developer Response

Acknowledged, `rebalanceFor()` does emit an event when called by `Compounder` but it would be fitting in `rebalance()`. Will look into events broadly as part of a future rewrite.

Final Remarks

Sickle rebalance strategy provides an efficient way of creating and self-managing Uniswap V3 pool positions. Each user can define their specific configuration for rebalancing. The protocol will also provide a bot that will trigger rebalancing on behalf of the user. The user also has the option to rebalance their position manually. The protocol earns fees from each rebalancing call but these fees can be bypassed by defining specific parameters. Rebalance parameters are calculated off-chain and provided to the user through the UI. The user must have particular knowledge and involvement on each call to bypass fees. The protocol should monitor these values and enforce paying fees if needed.