SOLIDIFIED

Audit Report for Longship - April 20, 2022

Summary

Audit Report prepared by Solidified covering the Longship AMM smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief was on 11 April 2022.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/shipyard-software/hydrofoil

Commit hash: `7facd7ce92b694cd3c07989274f10a938089d2e2`

```
contracts
├── Hydrofoil.sol
├── HydrofoilParent.sol
└── libraries
    ├── HydrofoilUtils.sol
    ├── InvariantCalcs.sol
    ├── SafeAggregatorInterface.sol
    └── SimpleDatabase.sol
```

Note on scope: The economic calculations have been verified against a provided specification. However, the correct functioning depends on external inputs, such as the regular update of the admin provided volatility parameter.

Intended Behavior

The smart contracts implement an automated market maker that allows liquidity providers to provide liquidity to a single asset pool, which can be used by traders to make two types of bets: leveraged long bets with a liquidation ratio and so-called repo bets.

SOLIDIFIED

Audit Report for Longship - April 20, 2022

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases have their limits and a remaining risk is present (see disclaimer).

Users of a smart contract 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**.

Note that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium-High	-
Code readability and clarity	Medium	-
Level of Documentation	Medium	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Longship - April 20, 2022

Issues Found

Solidified found that the Longship contracts contain 1 critical issue, 1 major issue, 2 minor issues and 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Hydrofoil.sol: bankruptcyLiquidateLong() can be called several times on a long position	Critical	Resolved
2	Hydrofoil.sol: Reentrancy may cause exploitable issues with ERC-777 tokens or other ERC-20 tokens with receive hooks	Major	Resolved
3	Hydrofoil.sol: Excess ETH sent to contract is not refunded and will be stuck in the contract	Minor	Resolved
4	Wide pragma range may cause integer overflows and compilation issues	Minor	Resolved
5	HydrofoilParent.sol: Missing zero check	Note	Resolved
6	HydrofoilParent.sol: Unused function	Note	Resolved
7	Hydrofoil.sol: Wrong message	Note	Resolved
8	Gas optimizations	Note	Resolved

SOLIDIFIED

Audit Report for Longship - April 20, 2022

Critical Issues

1. Hydrofoil.sol: bankruptcyLiquidateLong() can be called several times on a long position

In `bankruptcyLiquidateLong()`, only `removeLong()` is called and the long entry in the data-structure is not actually deleted. This could lead to a long being liquidated several times.

Recommendation

Add `delete(longs[nonce])`.

Major Issues

2. Hydrofoil.sol: Reentrancy may cause exploitable issues with ERC-777 tokens or other ERC-20 tokens with receive hooks

The function `unifiedTransmit()` is used in several places to send out ERC20 tokens or ETH. In these cases, the state is updated after the transfer occurs, which may lead to reentrancy vulnerabilities with ERC-20 compatible tokens that allow users to register receive hooks, such as ERC-777 tokens. A malicious user could exploit these receive hooks to eject code that performs a reentrant call.

To a lesser extent this issue also applies to ETH transfers. However, since the receiver is understood to be the wrapped ETH smart contract, there is currently no risk of a reentrancy attack unless an unconventional WETH version is used.

Recommendation

Consider using a reentrancy guard on external functions that use `unifiedTransmit()`.

Minor Issues

3. Hydrofoil.sol: Excess ETH sent to contract is not refunded and will be stuck in the contract

The function `unifiedTransmit()` is used as a unified way to receive assets. In the case of receiving ETH the contract assumes the underlying asset to be wrapped ETH and forwards the amount specified in the `howMuch` argument. Whilst there are checks in place to ensure that the amount of ETH supplied is sufficient, any excess sent by the users will be stuck in the contract.

Recommendation
Consider adding a requirement for `msg.value` to equal the `howMuch` argument.

4. Wide pragma range may cause integer overflows and compilation issues

The codebase allows for a wide range of Solidity compiler versions, including versions below 0.8.0. However, versions below 0.8.0 did not include automatic overflow protection which may lead to uncaught arithmetic errors.
In addition, it is not clear if the whole codebase is compatible with version 0.7.0.
Note, that `SimpleDatabase.sol` contains mathematical operations that overflow will not revert when compiled with these lower compiler versions. In such a case, a large enough `amount` that is passed to function `_mint()` will cause both `_balances[account]` and `_totalSupply` to overflow.

Recommendation
We recommend restricting the version pragma to 0.8.4 or above, since this version fixed important compiler bugs.

Informational Notes

5. HydrofoilParent.sol: Missing zero check

The function `adjustAddressChecker()` lacks a zero check on the address argument. This may lead to accidental misconfiguration.

Recommendation
Consider adding a check for address zero.

6. HydrofoilParent.sol: Unused function

The function `checkAddress()` is not used in the codebase anymore, since whitelisting has been replaced by admin-controlled deployment.

Recommendation
Consider removing unused code.

7. Hydrofoil.sol: Wrong message

In `bankruptcyLiquidateLong()` on line 663, the message when reverting is “Bankrupt”, but the cause for the reversion is that the system is NOT bankrupt.

Recommendation
Change message.

8. Gas optimizations

- `exp` is only used inside `setMultsForVolatility`, instead of converting 20 -> 16 -> 18 -> 16 decimals, one conversion could be saved (20 -> 18 -> 16) when directly passing 18 decimals to `exp`.
- Some functions could be declared as external to save gas (especially relevant for `InvariantCalcs` where large arrays are passed):
    - `HydrofoilUtils.exp`
    - `InvariantCalcs.getInvariant`
    - `InvariantCalcs.getInvariantsLP`
    - `InvariantCalcs.getInvariantsRepo`

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Longship or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Oak Security GmbH