Summary

Audit Report prepared by Solidified covering the Ola Finance smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief was on 11 October 2021.

The fixes were verified on 21 October 2021.

Audited Files

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

https://github.com/ola-finance/ola-audit
Commit hash: `3bd2e4e11e21945e2b47656f4b34cacdc12e2f34`
Commit hash for fixes: `a8ac3ff6085be7364cb25e9a0b0bdef1b833841a`

```
├── Core
│   ├── Interfaces
│   │   ├── EIP20Interface.sol
│   │   └── EIP20NonStandardInterface.sol
│   ├── LendingNetwork
│   │   ├── Comptroller
│   │   │   ├── ComptrollerInterface.sol
│   │   │   ├── ComptrollerStorage.sol
│   │   │   └── Unitroller.sol
│   │   ├── Delegators
│   │   │   ├── CErc20Delegator.sol
│   │   │   ├── ODelegator.sol
│   │   │   └── ONativeDelegator.sol
│   │   ├── ErrorReporter
│   │   │   └── ErrorReporter.sol
│   │   ├── OTokens
│   │   │   ├── CToken.sol
│   │   │   ├── CTokenInterfaces.sol
│   │   │   └── Maximillion.sol
│   │   ├── PriceOracle
│   │   │   └── PriceOracle.sol
│   │   └── versions
│   │       ├── 0.01
│   │       │   ├── CErc20V0_01.sol
│   │       │   ├── CEtherV0_01.sol
│   │       │   └── ODelegatorFactoryV0_01.sol
│   │       ├── 0.02
│   │       │   ├── CErc20V0_02.sol
│   │       │   ├── CEtherV0_02.sol
│   │       │   ├── ComptrollerV0_02.sol
│   │       │   ├── ODelegatorFactoryV0_02.sol
│   │       │   └── delegates
│   │       │       ├── CErc20DelegateV0_02.sol
│   │       │       └── ONativeDelegateV0_02.sol
│   │       └── 0.04
│   │           └── ComptrollerV0_04.sol
│   ├── Math
│   │   ├── CarefulMath.sol
│   │   ├── Exponential.sol
│   │   ├── ExponentialNoError.sol
│   │   └── SafeMath.sol
│   └── OlaPlatform
│       ├── Factories
│       │   ├── BaseMinistryFactory.sol
│       │   ├── IODelegatorMInistryFactory.sol
│       │   ├── PeripheryMinistryFactory.sol
│       │   └── deployers
│       │       ├── CErc20DelegatorDeployer.sol
│       │       └── ONativeDelegatorDeployer.sol
│       ├── InterestRateModels
│       │   ├── BaseImmutableJumpRateModel.sol
│       │   ├── BaseJumpRateModelV2.sol
│       │   ├── ImmutableJumpRateModel.sol
│       │   ├── InterestRateModel.sol
│       │   ├── JumpRateModelV2.sol
│       │   ├── LegacyInterestRateModel.sol
│       │   └── LegacyJumpRateModelV2.sol
│       ├── Registry
│       │   ├── Ministry.sol
│       │   ├── RegistryInterface.sol
│       │   └── RegistryStorage.sol
│       └── versions
│           └── 1
│               └── RegistryV1.sol
└── Peripheral
    ├── ComptrollerPeripherals
    │   ├── Bouncer
    │   │   ├── IBouncer.sol
    │   │   ├── WhiteListBouncer.sol
    │   │   └── deployers
    │   │       └── WhiteListBouncerDeployer.sol
    │   ├── IComptrollerPeripheral.sol
    │   ├── RainMaker
    │   │   ├── RainMakerInterface.sol
    │   │   ├── RainMakerStorage.sol
    │   │   ├── Reservoir.sol
    │   │   ├── SingleAssetDynamicRainMaker.sol
    │   │   └── deployers
    │   │       └── SingleAssetDynamicRainMakerDeployer.sol
    │   └── factories
    │       └── PeripheralFactory.sol
    ├── Lens
    │   └── CompoundLens.sol
    ├── PriceOracle
    │   ├── Band
    │   │   └── BandPriceOracle.sol
    │   ├── ChainLink
    │   │   └── ChainlinkPriceOracle.sol
    │   ├── interfaces
    │   │   └── IPriceOracle.sol
    │   └── open-oracle
    │       ├── OpenOracleData.sol
    │       ├── OpenOraclePriceData.sol
    │       ├── OpenOraclePriceOracle.sol
    │       ├── OpenOracleView.sol
    │       └── Uniswap
    │           ├── UniswapAnchoredView.sol
    │           ├── UniswapConfig.sol
    │           └── UniswapLib.sol
    └── Timelock
        └── Timelock.sol
```

Intended Behavior

The smart contracts extend the Compound protocol to add a few new features.

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	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the Ola Finance contracts contain no critical issues, no major issues, no minor issues and 8 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	Avoid shadow variables	Note	Resolved
2	ChainlinkPriceOracle.sol: Updated timestamp can be different	Note	Acknowledged
3	A mix of Solidity compiler versions	Note	Acknowledged
4	Function _acceptAdmin() checks for msg.sender != address(0)	Note	Resolved
5	CErc20Delegator.sol: Duplicate function implementation	Note	Resolved
6	Use named constants for exponent scale	Note	Acknowledged
7	CEtherV0_01.sol: Remove unwanted constructor	Note	Resolved
8	Miscellaneous notes	Note	-

Critical Issues

No critical issues found

Major Issues

No major issues found

Minor Issues

No minor issues found

Notes

1. Multiple contracts: Avoid shadow variables

Several contracts reuse state variable names for local variables. Though this is not harmful, it is recommended to not reuse the exact variable names and modify it to keep variable names unique.

- `RegistryV1.registerNewLnInternal()`: uint256 latestSystemVersion = latestSystemVersion;
- `CToken._reduceReservesFresh()`: olaReserveFactorMantissa
- `SingleAssetDynamicRainmake`: compAccrued

Recommendation
Consider making every variable name unique.

Update: Resolved

2. `ChainlinkPriceOracle.sol`: Updated timestamp can be different

The retrieve timestamp methods return only the updated timestamp and not the price updated. This prevents the caller from using an accurate timestamp if the price gets updated before this function is called.

Recommendation
Consider returning the price and timestamp in one function instead of using different ones.

Update: Response from the team - “As we do not use oracles’ update times within the contracts, we see no reason to provide a single getter for both timestamp and price. Additionally, in the Lens contract (used by our app and other off-chain services) we wrap the 2 getters into one.”

3. A mix of Solidity compiler versions

The codebase mixes code across two major compiler versions:

- `^0.5.16` - for Compound contracts
- `0.5.16` - for `IBouncer.sol` and `WhiteListBouncer.sol`
- `>=0.5.0` - for `IUniswapV2Factory.sol`
- `^0.7.6` - for newly written contracts

Recommendation
Consider using the same solidity compiler version for all contracts.

Update: Response from the team - “We prefer to maintain forked contracts with their original versions. New contracts that interact directly with forked contracts also maintain the same version. For independent new contracts that we write we use the latest solidity version.”

4. Multiple contracts: Function _acceptAdmin() checks for msg.sender != address(0)

The function `_acceptAdmin()` in `Unitroller.sol`, `Ministry.sol` and `SingleAssetDynamicRainMaker.sol` contracts checks for `msg.sender != address(0)` instead of checking `pendingAdmin != address(0)` as the comment above suggests.

Recommendation
Consider either removing the check or updating the code to check for `pendingAdmin != address(0)`.

Update: Resolved

5. CErc20Delegator.sol: Duplicate function implementation

The contract provides exactly the same implementation for functions already inherited from `ODelegator.sol` - `delegateTo()`, `delegateToImplementation()` and `delegateToViewImplementation()`.

Recommendation
Consider removing the duplicated functions implementation.

Update: Resolved

6. Multiple contracts in OlaPlatform: Use named constants for exponent scale

Various contracts in OlaPlatform use `1e18` for the exponent scale rather than inheriting `expScale` from the Compound code.

Recommendation
Consider replacing instances of `1e18` with a constant variable.

Update: Acknowledged

7. `CEtherV0_01.sol`: Remove unwanted constructor

The contract contains a constructor which is used to initialize the admin variable. Since the contract is only used via delegate calls, it can be removed.

Recommendation
Consider removing the constructor.

Update: Resolved

8. Miscellaneous notes

Following are a few generic notes to improve code quality and readability:

- `ChainlinkPriceOracle.sol`: Duplicate functions which do the same action. Consider merging `getPriceForAsset` and `getAssetPrice`.
- `CErc20V0_01.sol` - line #189 misspelled word in comment - `complaint` instead of `compliant`.
  Update: Resolved
- Consider using modifiers rather than copy pasting access control at the start of each function to improve readability.
  Update: Acknowledged
- General lack of reentrancy guards outside of `CToken`. Though there are no other possible reentrancy issues, consider adding the check to be extra careful.
  Update: Acknowledged
- Various contracts have a copy pasted assembly proxy, that proxy should check for the output pointer location rather than assume it's at `0x40`.
  Update: Acknowledged

Recommendation
Consider addressing the notes to improve code quality and readability.

Disclaimer