Cod3x LEND 2 - Beirao audit 11/10/2024

Introduction

A time-boxed security review of the Granary V2 protocol was done by Beirao, with a focus on the security aspects of the application's smart contracts implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About Beirao

I'm an independent smart contract security researcher. I extend my skills as a contractor specializing in EVM smart contracts security. If you're in need of robust code review, I'm here to help. We can get in touch via Twitter or Email.

About Cod3x Lend

The purpose of this audit is to highlight high level errors due to the new rehypothecation and minipool mechanism. This is not a line-by-line review, so some hidden errors may remain.

Observations

Cod3x lend is a AAVE V2 fork that adds:

- Updated pragmas (^0.8.0)
- Lending pool external rehypothecation
- The concept of minipools. Minipools are sub-markets that have a privileged borrowing capacity on the main lending pool.

Privileged Roles & Actors

Same as AAVE V2 roles.

Previous audits

Trail of Bits:

Cod3x Foundation - Cod3x Lend - Comprehensive Security Assessment.pdf

Beirao:

3e1284ef-7b6f-4ee5-ae1e-1ee05ce6a8a5_Cod3x_LEND_-_Beirao_audit_31082024.pdf

Zigtur:

Cod3x-Lend_Zigtur_Audit_V1.1.pdf

Severity classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - the technical, economic and reputation damage of a successful attack

Likelihood - the chance that a particular vulnerability gets discovered and exploited

Severity - the overall criticality of the risk

Security Assessment Summary

***review commit hash* -** 0f8a29b3

***fixes review commit hash* - ffffffff**

Deployment chains

- All EVMs.

Scope

The following smart contracts were in scope of the audit: (total : **8056** SLoC)

- `contracts/protocol/**`

Findings Summary

Summary :

- **1** Critical(s)
- **1** High(s)
- **0** Medium(s)
- **3** Low(s)
- **2** Informational(s)

ID	Title	Status
[C-01]	Main lendingPool liquidations breaks internal accounting due to rehypothecation	Fix
[H-01]	`protocol/configuration` is not mature enough ⇒ `Minipool` and `aERC6909` can't be updgraded	Fix
[L-01]	`deposit()` should check if the minipool has debts and repay it if necessary	Fix
[L-02]	Remove `recentBorrow` logic since it's not fully implemented and dangerous	Fix
[L-03]	`AERC6909.getIdForUnderlying()` is error prone	Fix
[L-04]	`_repayLendingPool()` can be safer and simplified	Fix
[I-01]	Move `MiniPoolCollateralManager` in the liquidation logic lib	Fix

ID	Title	Status
[I-02]	Change `ATokenERC6909` default symbole and name pattern to better match and add a restricted setter for `ATokenERC6909` name and symbol	Fix

Detailed Findings

[C-01] Main lendingPool liquidations breaks internal accounting due to rehypothecation.

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/core/minipool/MiniPoolCollateralManager.sol#L227

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/core/lendingpool/LendingPoolCollateralMana#L215

Severity

Impact: High, breaks liquidity accounting.

Likelihood: High, happens on every liquidations.

Description

When the `aToken` is repaid, `handleRepayment()` must be called to update the aToken's internal accounting. This function is not called on `lendingpool` and `minipool` liquidations. This will confuse the accounting of the `aToken` and `AERC6909`.

Recommendations

`handleRepayment()` must be added on liquidation logic. on lendingpool and minipool.

[H-01] `protocol/configuration` is not mature enough ⇒ `Minipool` and `aERC6909` can't be updgraded

https://github.com/Cod3x-Labs/Cod3x-Lend/tree/main/contracts/protocol/configuration

Severity

Impact: High, if a vulnerability is found in the minipool or AERC6909 implementations, the administrator will not be able to update + the admin address cannot be changed.

Likelihood: Medium, it can happen.

Description

- Lack of setter to change the admin in `MiniPoolAddressesProvider`.
- `LendingPoolAddressesProviderRegistry` can be removed since the main lendingpool should only be deployed once on a chain.
- add a `setAddressAsProxy` in `MiniPoolAddressesProvider` just like in `LendingPoolAddressesProvider`.
- `setMiniPoolImpl` and `setAToken6909Impl` should call `_updateImpl` instead of just updating the storage.
- add a `getMinipoolsList` getter.
- more generally cleanup everything and normalize typos between `LendingPoolAddressesProvider` and `MiniPoolAddressesProvider`.
- maybe remove the `_marketId` variable from `LendingPoolAddressesProvider`.

- cleanup the `MiniPoolAddressesProvider` storage.

[L-01] `deposit()` should check if the minipool has debts and repay it if necessary

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/core/minipool/MiniPool.sol#L126

Description

If a minipool has flow borrowed from the main lendingpool, the deposited funds will not repay the debt, but will simply be transfered to the aToken as usual.

Minipool debt from the main lendingpool should always be the first debt to be repaid.

Recommendations

Add `_repayLendingPool()` in `Lendingpool.deposit()`.

[L-02] Remove `recentBorrow` logic since it's not fully implemented and dangerous

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/main/contracts/protocol/libraries/configuration/UserRecentBorrow.sol

Description

The goal of the `recentBorrow` logic was to adjust the LTV and liquidation threshold depending on the collateral used. This logic was not fully implemented.

Recommendations

Regarding Zigtur issue M-04, I recommend removing `recentBorrow` from `BorrowLogic` and `FlashLoanLogic` on both `lendingpool` and `minipool`.

[L-03] `AERC6909.getIdForUnderlying()` is error prone

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/tokenization/ERC6909/ATokenERC6909.sol#L308

Description

`getIdForUnderlying()` doesn't have the same effect depending on whether the `asset` is `tranched` or not. If the asset is an aToken, the output is deterministic. If the asset is a classic ERC20, then the return IDs are the next correct IDs.

Recommendations

Create 2 different functions: one deterministic and one non-deterministic.

[L-04] `_repayLendingPool()` can be safer and simplified

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/core/minipool/logic/MiniPoolBorrowLogic.sol L325

https://github.com/Cod3x-Labs/Cod3x-Lend/blob/0f8a29b3daead4e73f09c2c827a6f237f664c2cf/contracts/protocol/core/minipool/MiniPool.sol#L268

Description

Sometimes the amount passed to `_repayLendingPool` is different from the actual amount repaid.

We can be improve `minipool._repayLendingpool()` by just removing the amount passed and always repay the maximum possible debt. (see recommendations)

Recommendations

https://gist.github.com/beirao/2e79c9c4a72a02cfb5498cbb95c4e9ce

[I-01] Move `MiniPoolCollateralManager` in the liquidation logic lib

[I-01] Change `ATokenERC6909` default symbole and name pattern to better match and add a restricted setter for `ATokenERC6909` name and symbol