



Exactly Smart Contract Audit



Exactly Protocol

March 2023

Smart Contract Audit

V230404

Prepared for Exactly Protocol • April 2023

1. Executive Summary

2. Assessment and Scope

First Stage

Second Stage

3. Summary of Findings

4. Detailed Findings

EXA-40 Fixed rate curve discontinuity

EXA-41 Interest rate model change enables retroactive fee collection

5. Disclaimer

1. Executive Summary

In January 2023, Exactly Protocol engaged Coinspect to perform a source code review. The objective of the project was to evaluate the security of the Interest Rate Model as well as other relevant parts of their protocol such as the implementation and changes introduced of the Markets.

In March 30 2023, Exactly Protocol engaged Coinspect to perform a source code review. The objective of the project was to evaluate the security of new commits that include fixes and modifications to the Market contract.

This report joins a timeline of correlative security audits along with issue IDs, highlighting multiple distinct bugs within the same scope. For more information on the results identified in our previous assessment on Exactly Protocol, please refer to “Coinspect - Smart Contract Audit - Exactly Protocol V221028” of October 2022.

The following issues were identified during the initial assessment:

High Risk	Medium Risk	Low Risk
Open 0	Open 0	Open 0
Fixed 0	Fixed 0	Fixed 1
Reported 0	Reported 0	Reported 1

Coinspect identified one low-risk and one informational issue. EXA-40 is an informational issue regarding a discontinuity in the interest rate model curve and EXA-41 shows how modifying the interest rate model could charge treasury fees retroactively.

2. Assessment and Scope

On the following document, only fixes and changes related to the **protocol** are addressed. Everything else regarding the RewardsController implementation is included under a separate report. This audit focuses on the fixes and changes introduced to the Markets and other core contracts such as the Interest Rate Model implementation.

First Stage

The audit started on **Jan 23, 2023** and was conducted on the **next** branch of the git repository at <https://github.com/exactly/protocol> as of commit [da9ddd7f847ebdad24c21fdae049f9d4523e9313](#) of Jan 24, 2023.

The audited files under the contracts directory have the following sha256sum hash:

```
1a652289b8c16eb9678cabb726013cf0fa7ffd3f59b8f35f7fb9fbf62898f8ca InterestRateModel.sol
```

Coinspect reviewed the changes to the Interest Rate Model (IRM) that modify how fixed and floating rates are computed. It is worth mentioning that the Exactly team stated that this implementation is intended to be deployed on the Optimism Chain, which is relevant due to the current gas consumption of the entire rewards logic (update, allocation, and distribution).

Regarding the changes introduced to the Interest Rate Model and how curves are calculated, Coinspect found that due to a discontinuity in the curve, the rate slightly decreases with an increase of the utilization, **EXA-40**.

Second Stage

The audit started on **March 30 2023** and was conducted on the **origin/coverage/protocol** branch of the git repository at github.com/exactly/protocol as of commits of the Market.sol file:

- [dd36c1c0e2cc6eae9aeb508c08c12fb89fa7368f](#)
- [783b0c351bae4c08f82d8175e2a224c5b254f84d](#)

On commit [dd36c1c0e2cc6eae9aeb508c08c12fb89fa7368f](#), a previously commented line was un-commented in `Market.sol` enabling deposit to treasury when changing the interest rate model contract between non-zero implementations. This ensures that the deposit will be made using the calculated values of floating debt up to that moment. Coinspect identified an issue where fees could be charged retroactively if the treasury fee rate is modified before changing the interest rate model, **EXA-41**.

The commit [783b0c351bae4c08f82d8175e2a224c5b254f84d](#) adds to the `Market.sol` implementation calls to the `RewardsController` update hooks when clearing bad debt and when calling `internalSeize`.

3. Summary of Findings

Id	Title	Total Risk	Fixed
EXA-40	Fixed rate curve discontinuity	Info	-
EXA-41	Interest rate model change enables retroactive fee collection	Low	✓

4. Detailed Findings

EXA-40

Fixed rate curve discontinuity

Total Risk

Info

Impact

-

Location

InterestRateModel.sol

Fixed

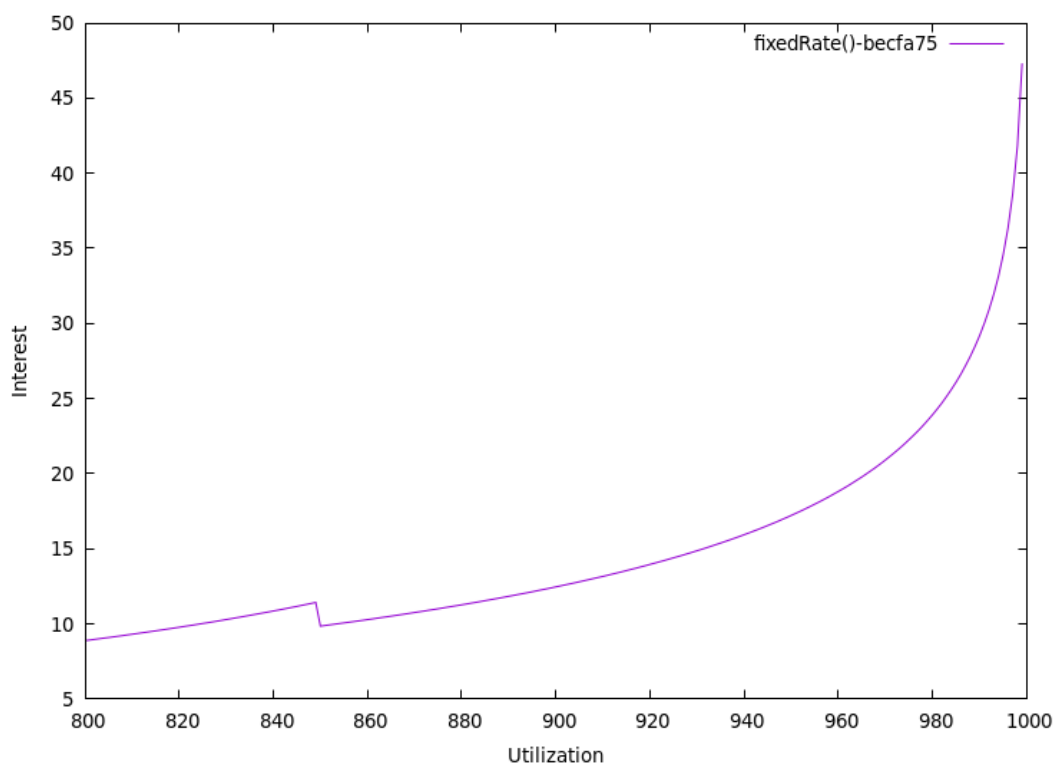
-

Likelihood

-

Description

Due to the switch from approximation to logarithm-based calculation of the integral, there is a discontinuity in the Fixed rate curve happening at the `PRECISION_THRESHOLD` point. This causes the calculated rate to slightly decrease with an increase of the utilization. In the following figure we can see this effect (greatly amplified for demonstration purposes):



Recommendation


Evaluate if, with the operating parameters selected, the precision of the current algorithm is acceptable.

Status

Acknowledged.

The Exactly Team stated: *“Using an approximation method is unavoidable due to a limitation in the precision of the logarithm function implemented. The chosen method guarantees that, under actual operating conditions, the discontinuity at the transition has an immaterial effect on the value of the resulting rate. Furthermore, the global error for the chosen method is much lower than that of other alternatives that guarantee continuity at the transition point.”*

EXA-41**Interest rate model change enables retroactive fee collection**

Total Risk Low	Impact Medium	Location Market.sol
Fixed 	Likelihood Low	

Description

It is possible to considerably increase the amount of fees deposited to the treasury when changing the interest rate model (IRM) if the treasury fee rate is modified before changing the IRM.

When changing the interest rate model of a market via `setInterestRateModel`, fees are deposited to the treasury:

```
/// @notice Sets the interest rate model to be used to calculate rates.
/// @param interestRateModel_ new interest rate model.
function setInterestRateModel(InterestRateModel interestRateModel_) public onlyRole(DEFAULT_ADMIN_ROLE)
{
    if (address(interestRateModel) != address(0)) depositToTreasury(updateFloatingDebt());

    interestRateModel = interestRateModel_;
    emitMarketUpdate();
    emit InterestRateModelSet(interestRateModel_);
}
```

If an `InterestRateModel` was previously set and no fees were transferred to the treasury before making the rate model change, modifying the treasury fee will lead to a retroactive calculation of the fees transferred to the treasury:

```
/// @notice Sets the treasury variables.
/// @param treasury_ address of the treasury that will receive the minted eTokens.
/// @param treasuryFeeRate_ represented with 18 decimals.
function setTreasury(address treasury_, uint256 treasuryFeeRate_) public onlyRole(DEFAULT_ADMIN_ROLE) {
    treasury = treasury_;
    treasuryFeeRate = treasuryFeeRate_;
    emit TreasurySet(treasury_, treasuryFeeRate_);
}
```

The following test shows the scenario mentioned before:

```
function testSetInterestRateModel_ChargingTreasuryFeesRetroactively() external {
    market.setTreasury(BOB, 0.1e18);
    market.setInterestRateModel(InterestRateModel(address(new MockInterestRateModel(0.1e18))));
    market.deposit(10 ether, address(this));
    market.borrow(1 ether, address(this), address(this));
}
```

```
vm.warp(365 days);

market.setTreasury(BOB, 0.5e18); // ===== COMMENT THIS LINE TO CHANGE THE SCENARIO =====
market.setInterestRateModel(InterestRateModel(address(new MockInterestRateModel(0.1e18))));

console.log("Treasury Balance: %s", market.balanceOf(BOB));
}
```

By commenting the mentioned line, the process has the following two outputs:

```
Modifying the treasuryFeeRate:
Treasury Balance: 49751243781094527

Without Modifying the rate:
Treasury Balance: 9910802775024777

Ratio = Changing Rate Scenario / No Rate Change Scenario ≈ 5
```

Recommendation

Deposit treasury fees before modifying the treasury fee rate.

Status

Fixed on commit [4b86c35114294f229bc388a34c730794e51a7804](#).

The `setTreasury` function now collects fees before making any changes.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.