Vulnerability disclosure 2023-06-05 (now patched)

Summary

- On April 27, 2023, a whitehat submitted a vulnerability report via Immunefi.
- The report showed that in markets with $0 in deposits for any of the assets, it was possible to manipulate utilization rate to points above 100%, increasing interest rates to extremely high levels.
- The core team identified the interest rate model contract (IRMV1) as the source of the vulnerability.
- A fix was coded into a new interest rate model contract (IRMV2) and deployed.
- Certora confirmed IRMV2 fixed the vulnerability and that no other exploit vectors were possible.
- No attacks were carried out, no user funds were lost, no protocol funds were compromised.
- The Silo protocol is now more secure and functioning normally.

Bounty and comments

We paid out a bounty of 100,000 USDC to the whitehat for finding a critical vulnerability. We appreciate the work of kankodu and invite all whitehats to sift through the code again and submit reports via our bug bounty program.

We are grateful for the Certora team, who reviewed the shipped fix (IRMv2) manually to confirm the vulnerability was fixed and made sure no other attack vectors were possible.

Although a security vulnerability was spotted, Silo protocol's code is now more battle tested than ever. The vulnerability is fixed and no one was affected by it. This incident provided our team with a great opportunity to review our codebase again internally, and then have it vetted by top-class security researchers.

Security has always been our utmost priority, and we are now more confident than ever in the resilience of the Silo protocol.

Chronology

- On April 27, a whitehat shared a concern that, in markets with $0 in deposits for any of the assets, users could manipulate utilization rate to points above 100%, effectively increasing interest rates to extremely high levels. We estimate that around 5%-10% of funds deposited in the Ethereum markets were at risk, while markets on Arbitrum remained immune given that all markets had deposits.
- Within a few hours of the report, the core team attributed the cause of the vulnerability to the interest rate model contract (IRMv1) and concluded that depositing liquidity into the at-risk markets would ensure an attack of the reported nature was unprofitable to carry out. Following this conclusion, liquidity was deployed to some markets.
- By May 1, the team had coded, tested and deployed a new interest rate model contract (IRMv2) that fixed the vulnerability.
- On May 2, IRMv2 was applied to all silos on Arbitrum (see transactions under resources below.)
- On Ethereum, given that the DAO owned functions controlling the interest rate model contract (IRMv1), we proposed the ownership transfer of certain smart contracts to a Safe controlled by the core team. The transfer of ownership was needed to allow the core team to deliver a fix immediately without making the attack vector public during the proposal voting duration and queuing (5 days total).
- The governance proposal was successfully executed on May 7, giving the core team admin functions to migrate all silos to use the new IRMv2.
- With control, IRMv2 was deployed and applied to all silos on Ethereum (see transactions under resources below).
- The core team performed several safety checks and confirmed that markets are now all functioning properly.
- On May 10, the Silo team asked the Cetora team to review the shipped code (IRMv2) and ensure that it fully fixed the vulnerability.
- On May 24, the Certora team sent us the following report (quote):
  - We've successfully created a rule (`interestNotMoreThenMax`) for the `interestRateModel` that proves that the interest amount can't be higher than the max limit, meaning that the bug fix covers the exploited scenario.
  - We've fixed the current rules to match the updated code as well.

- To our understanding, the fix indeed prevents the attack scenarios. The proposed fix keeps the interest rate safe and does not allow it to increase in an undesirable way. We could not find any other way to harm the protocol with the high utilization rate.

Details of Vulnerability

Background

Sending of ERC-20 tokens to a silo address - a regular ERC-20 transfer that does not trigger silo smart contract functions - caused the liquidity to increase in the market (silo). The transfer increases the liquidity that can be borrowed, however, it doesn't not increase the `totalDeposits` or mint any share tokens - which is an expected protocol behavior.

However, in the now-deprecated `InterestRateModel` contract (IRMv1), the utilization ratio was calculated as the ratio of `totalBorredAmount` to `totalDeposits`. Borrowing the liquidity that is directly sent to the silo contract could cause the utilization ratio to be greater than 100%. As a result, when the utilization ratio became greater than 100%, the `InterestRateModel` model treated utilization as critical, causing the interest rate to rapidly soar to astronomical levels, consequently increasing the collateral value by multiple times. Interest that has not yet been repaid is also taken into account in collateral calculations.

The above conditions allowed collateral manipulations.

Attack example (POC)

- Assuming the amount of ETH deposits in the wBTC silo is zero, an attacker deposits and mints $1e^5$ *wei* of WETH collateral tokens in the isolated market.
- The attacker spends 1 ETH by sending it directly to the wBTC silo contract address.
- Using another wallet, the attacker makes a wBTC deposit that is enough to borrow the donated 1 ETH.
- In the next block, after 10 seconds, due to `totalBorrowedAmount` being extremely higher than `totalDeposits`, >5000 ETH of interest gets accrued. This makes $1e^5$ of `collateralTokens` associated with WETH that the attacker initially deposited worth >5000 ETH.
- Then attacker borrows 150,000 worth of XAI because the protocol values the attacker's collateral as ~5000 ETH while he only spent ~ 1ETH.

Description of the vulnerability fix

With the interest rate model contract (IRMv1) being the source of the vulnerability, the team made one main code change that changed the utilization ratio formula such that utilization ratio cannot exceed 100%. With the change in effect, even when the utilization ratio remains at maximum 100% for a significant period of time, the protocol functions as expected.

Along with changing the utilization ratio formula, we capped the interest rate at 10,000% per year. In addition, we adjusted the interest rate model configs to reduce the speed of interest rate growth in critical utilization regions. As a result, when the interest rate reaches the cap, the state of the model resets, causing the interest rate to drop from 10,000% per year to ~100% per year, allowing the isolated market to recover.

With these changes, the attack vector is not possible anymore. Borrowing the liquidity that is simply sent (donated) to a silo becomes pointless, because now it never forces the utilization ratio to increase above 100%.

All changes are shipped in a new interest rate model contact dubbed as IRMv2.

Breakdown of Changes

1. Utilization ratio calculations fix, 100% is a limit in `InterestRateModel` contract

Limiting the maximum utilization ratio to 100% allows the market to perform better interest rate computations without impacting any other part of protocol.

2. Limit for `rcur - RCUR_CAP` (FE/integrations, does not affect core protocol)

This is the limit for the current interest rate. We picked 10,000% of interest per year. The interest rate model works normally with a capped maximum value.

The protocol doesn't read the current interest rate of a certain token asset but rather uses the compounded interest since last update. However, the interest rate is used in the UI and external protocol integrations (for example, investing strategies).

3. Limit for `rcomp := RCOMP_CAP * _l.T.`

4. If we get a limit for `rcomp`, we reset `Tcrit` and `Ri` model parameters to zeros

Resetting parameters will make interest rate drop from 10,000% per year to 100% per year, at which point it will start growing again. Resetting parameters to zero give the market a chance to recover its state organically.

5. `beta` is divided by value of 4 for all IRM configs, except `stableLowCap`, `stableHighCap` and `bridgeXAI`

With current values of `beta` parameter, volatile assets will get their interest rate (proportional term) multiplied by 2 in one hour. Division of `beta` coefficient by 4 will result in changing time to increase from one hour to four hours, which will make the interest rate model behavior less aggresive.

If we forget about the integral term (which will have less impact in first hours of a critical utilization state), the proportional term will grow linearly. It will double in the first 4 hours, triple in 8, quadruple in 12, etc.

Resources

Silo formal verification rules and reports (including IRM V2)

Certora prover results for IRM V2. Shows that this attack can not be performed anymore for the fixed interest rate model version.

Certora prover violation for IRM V1. Prover counter-example for the deprecated code as the vulnerability verification.

IRM V2 deployments

Transactions updating Ethereum markets with IRMV2 on Ethereum:

https://etherscan.io/tx/0x13aaa2b0c9a57f2440e63d1201bedbaffbca01bec3dd743f2354d485a933ed51

https://etherscan.io/tx/0x0d7bcf54d60dbb42e3a2f48b06c91ff698f2cc1fee4dcae2dabc515b69109392

https://etherscan.io/tx/0x5113a786082738d413eba01d6c9c0d76b67611ebdb87ea954fa8a4854232b3c8

https://etherscan.io/tx/0x93ccedff30080b08d86c16991a6900121eb5eea1fec0fcd48efc76ceb2144c7c

https://etherscan.io/tx/0x04abfcdc19efeceabb50ea323e43cff2b679a98218264ac0b2045b5dae4f62b8

https://etherscan.io/tx/0xb85ad7c8ab2369bddf037ec9abb58f71b3d2e3d5f5385a3ee3f97ab48b436b43

https://etherscan.io/tx/0xbdc237bf3a38d6de87daa4214c49ffe7292e1f847f7cc01f3721083625d247be

https://etherscan.io/tx/0x0871362270e68fbd09096206a077fd5b0027044e412abc3bc10254c19f3bfe18

https://etherscan.io/tx/0xd87a631d32028d19c807599e56ea37f7ca7e5989898505f42ef2a0b21da4e40c

https://etherscan.io/tx/0xcfe0b35a7b296ed73a3e77b1bb5076483aeefe3dab60bd7edff7cd2d2e1b1dd6

Transactions updating Arbitrum markets with IRMV2 on Ethereum:

https://arbiscan.io/tx/0x081ab7a569d704d44a7332f0e8c104d1e0ee50e69a884b65af504d706282820e

https://arbiscan.io/tx/0x8a0d660502fcd85bd1fed9b4987c4be36496bf66382588611c95b31317cb74a9

https://arbiscan.io/tx/0xd3498c3277310f468c70fbc06f30ffb18d2c2a91e217a8d4b00bb431ce1d916b