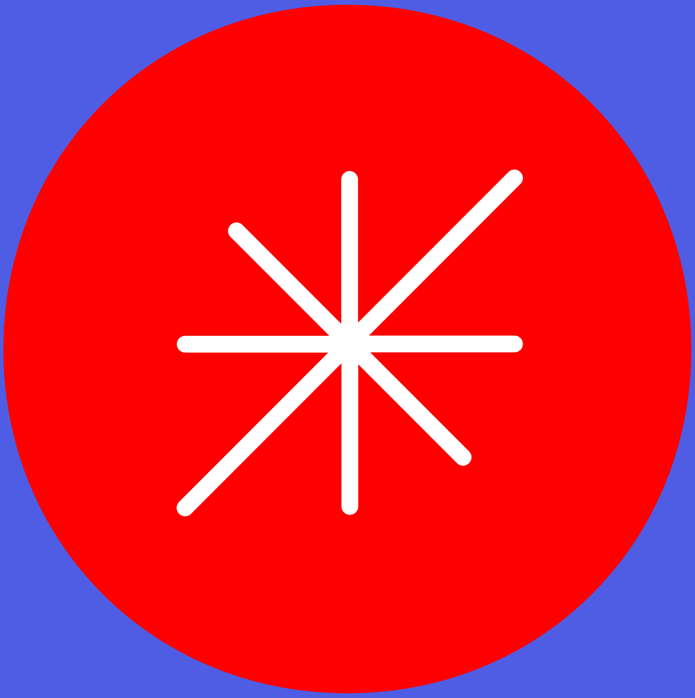
OpenZeppelin | security

Gamma Security Review



January 23, 2024

Table of Contents

Scope

The scope of the work performed as part of this engagement consisted of the following:

- Analysis of the hack from January 4th, 2024 and identification of the root cause of the vulnerability in `Clearing.sol` that led to the exploit
- Validation of the attack scenarios with updated parameters and confirmation of the exploit via repeatable tests
- Review and assessment of the new changes implemented in `ClearingV2.sol` addressing the root cause of the exploit
- Fuzzing testing scenarios and simulations of the updated `ClearingV2` contract (with the assumption that no asynchronous deposits are in place)

The security review consisted of two parts; the initial analysis covered the `contracts/Clearing.sol` file on the main branch of the OZ-Audit-Branch repository at commit 588ef85.

The second part of the review covered the `contracts/ClearingV2.sol` file on the sync-uniproxy branch of the OZ-Audit-Branch repository at commit 7d867b8.

System Overview

Gamma protocol serves as a concentrated liquidity management tool, enabling users to deposit liquidity into Gamma vaults. In exchange, stakers are minted shares and can earn a portion of the fees accrued by the protocol. These vaults are non-custodial and manage user funds via strategies. Currently, there are three types of strategies Gamma runs:

- **Dynamic Range (Wide / Narrow):** In both, liquidity ranges are automatically rebalanced and accrue fees that are compounded back into the position regularly. The wide-range strategy is for long-term liquidity providers that will want to have a lower impermanent loss in high-volatility environments. While the narrow-range strategy is for short-term liquidity providers and will perform better to earn higher fees in low-volatility environments.
- **Stable:** This strategy is used for stablecoin pairs where volatile pairs have wider ranges and more stable pairs have narrower ranges.
- **Pegged Price:** In this strategy assets used are staked tokens. The liquidity is provided around the net asset value of the provided asset and rebalanced upon price targets. It can go out of range with market volatility and thus not earn fees.

The Attack & Trust Assumptions

Originally, the vaults had two security mechanisms in place inside of the `Clearing` contract. First, each deposit was required to adhere to a strict ratio of two tokens, determined by the `priceThreshold` value. Second, deposits needed to align closely with the time-weighted average price (TWAP) of the UniswapV3 pools. Despite these measures, three key issues enabled the attacker to circumvent these security protocols:

- **Share Issuance:** The protocol mints shares with dependency on the price reported from the UniswapV3 pool, which makes the protocol vulnerable to price manipulation attacks if there are no protections in place.

- **`getDepositAmount`'s logic flow:** A branch of the `getDepositAmount` function permits attackers to deposit liquidity in any ratio, a critical flaw that can be exploited for malicious purposes.
- **Misconfiguration of `priceThreshold`:** The `priceThreshold` parameter was incorrectly set, leaving the system vulnerable to exploitation.

After a thorough investigation, it was confirmed that the attacker organized a complex attack on the system which used a combination of various issues in the codebase:

- **Manipulation of Pool Prices:** The attacker was able to manipulate the price of the underlying UniswapV3 pool that Gamma vaults based their share minting off of, and bypassed both the price ratio check to the maximum amount of TWAP price deviation limitation allowed, which led to an extremely deflated price for `token1`.
- **Exploitation of Vault Share Mechanism:** Taking advantage of the deflated `token1` price, the attacker was able to deposit large amounts of `token1` and mint vault shares at a low price.
- **Profiting from Redeemed Shares:** After redeeming the minted shares, the attacker made a profit at the vault shareholders' loss.
- **Repeated Arbitrage and Attack Cycle:** This was when the attacker arbitraged the pool composition back and repeated the attack again, after executing this process multiple times, it led to a loss of a significant amount of liquidity from multiple Gamma vaults.

Our team used a fork test file from DeFiHackLabs and modified it to test the following scenarios:

- The original attack parameters on the vulnerable vault. This was used for analysis of the root cause of the attack.
- The original attack parameters on a properly adjusted vault. Due to the `priceThreshold` check this reverts but does not show if adjusted attack parameters can still work
- Modified attack parameters on a properly adjusted vault. This reverts due to an improper ratio. Since the attacker can not get the quantity of `token0` to equal 0 with their swap without pushing the price too far and causing a revert on price change overflow. Allowing them to enter the vulnerable logic branch of the `getDepositAmount` function and make a one-sided deposit, so, it will revert.
- Modified attack parameters on a properly adjusted vault but `zeroDeposit` is set. This succeeds since the attacker does not need to get the quantity of `token0` to equal 0 to pass the improper ratio check. However, they still need to worry about not reverting on price change overflow.

The testing above, further validated the root cause of the hack was due to a vulnerability within `Clearing.sol`.

Mitigation Review

The Gamma Strategies team updated the vulnerable `Clearing` contract to `ClearingV2` and addressed the issues by including the following fixes:

- Added a new mechanism to check if the underlying pool tick is within the expected base range by the vaults.
- Corrected the flow of the `getDepositAmount` function. In the new implementation of the function, liquidity manipulation attacks cannot lead to bypassing the ratio check of the deposit.
- Correctly set the `priceThreshold` parameter.

The updated version of the `Clearing` contract, now known as `ClearingV2`, introduces an additional security layer compared to its predecessor. It includes checks to ensure the underlying pool price lies within defined base ticks. This new layer is an effective defense against price manipulation, and when combined with the previous two defense mechanisms, the TWAP price check and the deposit ratio check, it provides robust protection against attacks similar to the one that occurred on January 4th, 2024.

Testing Results

In addition to manual review, we also implemented fuzzing tests for two different versions of the `ClearingV2.sol`, including both synchronous and asynchronous minting methods. To utilize the fuzzer and test the system to a greater extent to find potential attack vectors, we configured the system settings to allow for a ±10% variance of the appropriate settings that were provided to us by the Gamma team. This fuzzing is done for 3 different pairs of volatile, stable and LST pairs.

Each of the pair tests performs the same sequence of transactions:

1. Swap `token1` for `token0` in the Uniswap-V3 pool.
2. Try to deposit into the system via `Uniproxy.sol`.
3. Withdraw from the system via `Uniproxy.sol`.
4. Finally swap back `token0` for `token1` in the Uniswap-V3 pool.

After all these steps, key details are logged into a Markdown file, and the tests check if the process is profitable. Additionally, the tests verify that when a user calls `UniProxy.deposit()` and it reverts, it is failing properly and is logging information about why it reverted.

The results of these fuzzing tests did not find any way to manipulate the system and profit from it. However, this degree of testing should be pushed further and refined with more time.

Full fuzzing test results as well as documentation related to the exact setup are included in the Gamma Fuzzing Github repo.

Additional Considerations

While analyzing the codebase, the team has found a couple of issues that might not cause a threat on their own, but are noteworthy. The newly added defense mechanism in `ClearingV2.sol` (which checks if the current tick of the underlying pool is in an expected range) poses some new restrictions on the system:

- An attacker can potentially manipulate the system to force a deposit to fail by moving the underlying pool's tick out of the expected range. While this leaves the pool in a position where arbitrage can be profitable, it provides a pathway for an attacker to perform sandwich attacks.
- This mechanism is similar to the existing TWAP price check to a certain extent as they both restrict the price movement of the pool before clearing a deposit request. While in theory, this is relatively safe since they work on top of each other, they might place a sufficiently strong restriction on the deposit, which could lead to a revert of a legitimate use cases depending on the pool pair volatility, actual settings of the base tick range and the `_priceThreshold` parameter.

Conclusion

On January 4th, 2024 Gamma Strategies experienced a hack due to the vulnerability within the `Clearing.sol` contract. As part of the two-week engagement, the OpenZeppelin team performed a root-cause analysis of the hack and confirmed that the successful attack was due to the attacker abusing the `getDepositAmount()`'s logic flaw to bypass the ratio checks.

The Gamma team subsequently shared an updated `ClearingV2.sol` contract, addressing the main root cause of the hack. Throughout the manual review of the codebase as well as fuzzing tests and simulations, the OpenZeppelin team was not able to successfully exploit the updated codebase, following the attack patterns from the previous hack. The work performed by OpenZeppelin was intended to assess whether the updated `ClearingV2.sol` contract is still susceptible to the attack pattern from the latest hack and does not provide any reassurance on the state of the security of other system components or the protocol in general.