Security Audit – Mango v3

Neodyme AG

May 17, 2022

(amended July 15, 2022)

Contents

Introduction

(Note: This report was amended on July 15, 2022 to reflect a fix for a low-severity issue.)

Mango engaged Neodyme to do a detailed security analysis of their on-chain mango-v3 smart contract. A thorough audit was conducted between January and April 2022.

The audit revealed one critical vulnerability, one high-severity vulnerability, one medium-severity issue, one low-severity issue and one informational finding. Mango has released a fix for all issues except the informational finding.

In this report, we outline the most relevant findings of our audit.

Project Overview

Mango combines multiple De-Fi features into a single smart contract and thereby enables users to cross-collaterize their different assets into a single Mango account. Specifically, Mango offers functionality to lend, borrow and swap on Serum DEX and trade leveraged perpetuals.

Each user has a limited basket of active assets. The user can deposit and withdraw assets at will. Each asset is handled by a corresponding RootBank that keeps track of the value of lent and borrowed positions of that asset and which must be updated frequently. By default, if the user deposits assets and does not employ them in any way, these assets are considered part of the lending pool and thus earn interest for the user. Vice versa, the user can, of course, borrow assets. However, the health of a user's account must always remain positive. The health is calculated by summing over the user's deposits and borrows, their assets held in spot markets, and their perp positions. Specifically, it is the weighted sum of all owned assets minus the weighted sum of all liabilities[1]. The asset and liability weights vary depending on the underlying asset. They also encode the initial margin and the maintainance margin, the former setting a condition for opening new positions using the account and the latter for liquidating the account, as is standard.

The current value of a specific asset is taken from a MangoCache account which stores all relevant value data as given by the current lending and borrowing weight from the root banks, the current asset prices as given by on-chain oracles (namely Pyth or Switchboard), and by the perp market funding rates. All relevant operations which depend on the value of an asset – such as anything that may change the health of a Mango account – ensure that the data they source from the MangoCache is up-to-date.

More detailed information on Mango can be found in its documentation.

[1] https://docs.mango.markets/mango/health-overview

Scope

Scope of the audit was the mango-v3 smart contract, which can be found at (https://github.com/blockworks-foundation/mango-v3). The last audited commit hash was 3583fa19a909aaa4113bcdb23b35c5bede6866ae .

Methodology

Neodyme's audit team, which consists of security engineers with extensive experience in Solana smart contract security, reviewed the code of the on-chain contract, paying particular attention to the following:

- Ruling out common classes of Solana contract vulnerabilities, such as:
  - Missing ownership checks,
  - Missing signer checks,
  - Signed invocation of unverified programs,
  - Solana account confusions,
  - Re-initiation with cross-instance confusion,
  - Missing freeze authority checks,
  - Insufficient SPL token account verification,
  - Missing rent exemption assertion,
  - Casting truncation,
  - Arithmetic over- or underflows,
  - Numerical precision errors.
- Checking for unsafe design that might lead to common vulnerabilities being introduced in the future,
- Checking for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain,
- Ensuring that the contract logic correctly implements the project specifications,
- Examining the code in detail for contract-specific low-level vulnerabilities,
- Ruling out denial-of-service attacks,
- Ruling out economic attacks,
- Checking for instructions that allow front-running or sandwiching attacks,
- Checking for rug-pull mechanisms or hidden backdoors,
- Checking for replay protection.

Findings

All findings are classified in one of four severity levels:

- **Critical**: Bugs that will likely cause a loss of funds. This means that an attacker can trigger them with little or no preparation or even accidentally. Effects are difficult to undo after they are detected.
- **High**: Bugs which can be used to set up a loss of funds in a more limited capacity, or to render the contract unusable.
- **Medium**: Bugs that do not cause a direct loss of funds but lead to other exploitable mechanisms.
- **Low**: Bugs that do not have a significant immediate impact and could be fixed easily after detection.

Name	Severity	Status
Orderbook sides were not sufficiently verified	Critical	Resolved
Perp value calculation can overflow	High	Resolved
Missing vault check in resolve_token_bankruptcy	Medium	Resolved
Deposit does not trigger interest update	Low	Resolved
RootBankCache timeout inconsistency	Informational	Acknowledged

Orderbook sides were not sufficiently verified (Critical; Resolved)

Severity	Impact	Affected Component	Status
Critical	Likely loss of funds	Perp orderbook	Resolved

When trading Mango perpetuals, one places orders on the corresponding orderbook consisting of a bids and an asks account. When loading the book, Mango hat two checks in Book::load_checked:

```
Ok(Self {
    bids: BookSide::load_mut_checked(bids_ai, program_id, perp_market)
        ?,
    asks: BookSide::load_mut_checked(asks_ai, program_id, perp_market)
        ?,
})
```

and in BookSide::load_mut_checked:

```
check!(account.owner == program_id, MangoErrorCode::InvalidOwner)?;
let state = Self::load_mut(account)?;
check!(state.meta_data.is_initialized, MangoErrorCode::Default)?;
match DataType::try_from(state.meta_data.data_type).unwrap() {
    DataType::Bids => check!(account.key == &perp_market.bids,
        MangoErrorCode::Default)?,
    DataType::Asks => check!(account.key == &perp_market.asks,
        MangoErrorCode::Default)?,
    _ => return Err(throw!()),
}
```

There is a missing check for perp_market.bids == bids_ai.key and perp_market.asks == asks_ai.key, as BookSide::load_mut_checked only checks for internal consistency of the state. That way an attacker could supply the asks account as bids and vice versa, allowing to buy the entire bid-side and selling it to the asks side.

Fix With commit 022caa5474446bb8a3ccdf850334b44691346011 Mango quickly responded and deployed a fix. Neodyme verified this fix.

Perp value calculation can overflow (High; Resolved)

Severity	Impact	Affected Component	Status
High	Likely loss of funds	Health calculation	Resolved

When calculating the health of an Account, Mango also must calculate the value of a perp position. This is done through the function get_val which contains the following code snippet:

```
        let bids_base_net = self.base_position + self.taker_base + self
            .bids_quantity;
        let asks_base_net = self.base_position + self.taker_base - self
            .asks_quantity;
```

Here, `self` is of type `PerpAccount` and `self.bids_quantity` equals the amount of open bids on the order book. When calling place_perp_order, the `self.bids_quantity` is user-controlled and can be set to an arbitrary value, triggering an overflow of bids_base_net. Vice versa, the same is possible for asks_base_net. This overflow can corrupt the asset value calculation so that base and quote both get rated with positive value. Ignoring the weighting, these should normally cancel out. The overflow leads to a much higher rating of the perpetual and therefore results in a loss of funds when using this wrong overvalued health for borrowing uncovered assets.

Fix Mango quickly fixed the overflow in commit b0d239d7f721371677e81eb4df1c89dfd7d6b70b by using checked math. Neodyme verified this fix.

Missing vault check in resolve_token_bankruptcy (Medium; Resolved)

Severity	Impact	Affected Component	Status
Medium	Duplication of insurance_transfer	Health calculation	Resolved

If an account still has liabilities but no more assets for liquidators to take, the account is bankrupt and the mango insurance fund starts paying. This process is implemented through resolve_token_bankcruptcy and has to be called by a liqor, who gets a premium for repaying. As the insurance fund only holds the quote currency (USDC), the liqor gets the corresponding liability value in USDC and repays the bankrupt account in the asset from his own balance.

The process is implemented as follows:

1. The insurance fund SPL-transfers the corresponding USDC value from the insurance fund vault to the Mango quote vault
2. The new quote balance gets credited to the liqors Mango Account
3. The liqor repays the bankrupt account in assets from his Mango Account

Unfortunately, Mango was missing a check in step 1 for the quote vault. This means that a malicious liqor could supply his own token account as quote vault and get credited twice (in SPL and Mango Account) but only repay once.

Fix Mango quickly fixed the missing check in commit 55bc7246ce2b0cefc46b27aaddd43dc0d32b04dc by adding a check for quote_vault_ai.key == &quote_node_bank.vault. Neodyme verified this fix.

Deposit does not trigger interest update (Low; Resolved)

Severity	Impact	Affected Component	Status
Low	Receive interest without providing assets	RootBank updates	Resolved

The borrow index, which is used to keep track of the lending interest, is stored in the RootBank of each asset and can be updated by anyone. It is possible to deposit an asset, immediately update its root bank, then withdraw the asset plus the interest of the deposit for the time frame since the root bank was last updated. Thus, the attacker is effectively receiving interest from the contract without actually providing the asset.

If the attacker has some other short-term way of using the asset, this gives them a financial advantage. One obvious way to do this is by using another lending market X that also has no origination fee. The attacker deposits their money in X and then can do the following:

1. Borrow asset A from X, deposit it into mango, update the root bank for A, withdraw it immediately and repay the borrow from X
2. Borrow asset B from X, deposit it into mango, update the root bank for B, withdraw it immediately and repay the borrow from X
3. …

This enables the attacker to get the sum of the APYs of all assets on mango's lending market without actually providing any of the assets.

Fix Mango added an interest update in the deposit instruction on June 24, 2022 with commit `bbae354a678d2fccc323aefa05ae1f2c267b69f8`. Neodyme verified the fix. The delay was due to the low severity of the issue and the need for clients to be modified for this fix.

RootBankCache timeout inconsistency (Informational; Acknowledged)

Severity	Impact	Affected Component	Status
Informational	No real impact	Cache	Acknowledged

Because of Solana's limit on accounts per transaction, Mango makes use of a caching mechanism for storing important global data. One of those is the borrow index of RootBank, which tracks the collected interest rate over time. To guarantee the integrity of cached data, Mango checks that last_update is not beyond the specified timeout. The last_update property of the RootBankCache gets updated in cache_root_banks, setting it to the current timestamp and setting the cache[root_bank].indices to root_bank.indices. Hereby, the root_bank indices are actually cached again over the corresponding NodeBanks through update_root_bank. Mango has no check in cache_root_banks so that, now_ts − root_bank.last_updated < Timeout holds. This means that one could update the index of a NodeBank without calling update_root_bank afterwards, resulting in a non-updated RootBank with an old last_updated timestamp, but cache_root_banks could still be called and the integrity of the Cache will be seen as valid. From an attacker's perspective, this means, that there could be free interest lying around. However, the interest over short periods is probably negligible.

One must note that such an event is very unlikely to happen, as Mango operates crankers calling these functions regularly. Additionally, the Mango web interface also calls them when emitting a transaction. Furthermore, there is an incentive for current lenders to update the RootBank, as they want to get their interest paid.

Considering all this, Mango acknowledged the inconsistency and accepts the very low risk of this happening.

Neodyme AG

Dirnismaning 55
Halle 13
85748 Garching
E-Mail: contact@neodyme.io

https://neodyme.io