Meteora Dynamic Vaults - The Yield Layer for Solana

Andrew Nguyen, Yee Guan Tian, Dionis Chiua, Meow
Meteora

WIP, Updated Feb 2023

1 Introduction

Users enjoy custodial wallets offered by centralized cryptocurrency exchanges for its convenience. However, they have no control over their funds and it is not always transparent what these businesses do with their assets. Holding assets in decentralized wallets is powerful, as it allows one to have full control over their funds. However decentralized finance (DeFi) is fraught with challenges; earning yield on their assets exposes the funds to risks such as hacks or rug-pulls when they deposit into another protocol. Yield also varies greatly between platforms; constant monitoring and rebalancing of funds is needed to optimize returns.

It is not possible for the average user to stay ahead of yield changes and perform risk management all the time, automation is needed. However, building this automation is non-trivial; it involves integration with multiple yield generating platforms and executing allocation strategies that protect liquidity while trying to earn the best-case yield. There are also other factors to consider, such as fund accessibility and mitigation of risks of connected protocols.

In this paper, we present Meteora Dynamic Vaults - The Yield Layer for Solana. This comprises an end-to-end risk management framework of optimising yield, mitigating lending protocols risks, and maintaining full principal liquidity.

We also introduce Hermes - our vault keeper program, which consistently monitors annual percentage yield (APY), amount of reserves available and utilization rate of each and every lending pool that user funds are deposited in, rebalancing between them for optimized yield while preserving liquidity.

The benefits of the vaults are not limited only to users, but also extends to that of decentralized applications (dApps), decentralized autonomous organizations (DAOs) and protocols, where their treasuries can be sizeable. The vaults have been designed to be easily integratable through a straightforward SDK, forming a layer for anyone with passive liquidity to connect to.

We believe that the vaults can serve a critical role as the yield layer for all of Solana, where users and protocols can connect easily with lending protocols, enabling all assets in the ecosystem to grow and earn yield safely while preserving liquidity.

2 Key Challenges & How Meteora Solves Them

We will examine the key challenges that Meteora Dynamic Vaults are designed to solve.

2.1 Challenges faced by users

Users are unable to consistently monitor their funds 24/7. Funds are not optimized for yield as they do not always have all information at hand, to determine which lending protocols will give them the most optimal yield. When a black swan event happens, they cannot react fast enough to withdraw funds, especially if it happens during the time they are offline.

2.2 Challenges faced by protocols, wallets and treasuries

Like users, assets kept on protocols, wallets and treasuries are generally not optimized due to the difficulty of aggregating the most optimal yield. This leads to them losing out on opportunities to generate more yield for the assets. Protocols also need to rely heavily on giving out their own tokens for liquidity mining (LM) rewards to attract liquidity providers (LPs), which is not sustainable in the long run.

Designing their own yield aggregator and monitoring system that optimizes yield while keeping the funds safe, involves many parameters such as integration and lending protocol assessment. This is resource intensive and time consuming to build, and not the core competency of this category of liquidity holders.

2.3 How Meteora Solves The Challenges

Meteora comprises an end-to-end risk management framework of optimising yield, mitigating lending protocols risks, and maintaining full principal liquidity. The vaults have done the work of integrating with lending protocols and their lending, allowing for real-time yield optimization. We now have over 50 lending reserves connected, across 6 protocols at the time of writing.

Our keeper program is designed to search for the best yield amongst connected lending protocols and rebalance allocations across them. To keep funds safe and maintain liquidity of deposited amounts, it monitors 24/7 for lending pool utilization rates and reserves levels of the protocols, withdrawing funds whenever the predetermined thresholds are reached.

Through maximum allocation, determined via a security matrix covered in section 4.3.3, Meteora manages risk across lending protocols by ensuring that the vault allocations are spread out.

To extend these benefits to advanced users, Meteora comes with a straightforward SDK and library of pre-built modules and code samples, for rapid app development and plug-and-play.

3 Design Goals

The goal of Meteora Dynamic Vaults is to solve the problems discussed above. Design principles include:

3.1 Security and safety of principals

Principals are safe at any given point in time; they can only be deposited into trusted and decentralized protocols for yield generation. The keeper program only stores the logic to find optimal yield allocations and limits the fund flows from the vaults to the protocols, it is unable to access the funds or claim principals. We seek to upgrade the authority for decisions around lending protocols integration and allocations to the decentralized Autonomous Organization (DAO).

3.2 Full liquidity at all times

Deposited assets must be liquid and accessible at all times, where users can withdraw funds at will. The vault's total deposit amount is always checked against the amount of reserves left in the lending platform; if the liquidity reserve in the pool is less than the predetermined threshold, the vault will proceed to withdraw from the pool to ensure that there will be sufficient liquidity for user withdrawals.

3.3 Most optimized yield returns

Vault program must be able to monitor and calculate yield variations across all connected lending platforms, and dynamically allocate and rebalance assets to the one with the highest returns. Annual percentage rate (APR) of a lending pool depends on various factors - borrowing amount, depositing amount and the interest rate model. Deposit APR decreases when we deposit in a lending pool because the borrowing interest is shared with other depositors. The algorithm to find optimal yield allocation must be able to compare and find the best APR amongst the platforms for a given deposit sum.

3.4 Ease of integration and usage

The vaults and SDK needs to be straightforward and easy for any users or protocols, to utilize and build a range of applications on our system. This includes full guides, code examples and an API to help anyone connect to the vaults and gain access to all integrated lending reserves easily. We want to make the vaults the yield infrastructure for all of Solana.

3.5 Event monitoring and tracking

Lending pool APY, utilization rates, reserve levels are monitored continuously for better yield opportunities and risk management. Solana's composability, speed and low transaction fees provide on-chain event monitoring benefits that exceed any other blockchain, and allows us to achieve the design principles set out above.

4 System Design

We begin with an overview of the vaults program, alongside with the key components and their detailed working. We will introduce our rebalance crank mechanism as well as our design for Sandwich Attack prevention. Next, we will do a deep dive into Hermes - our vault keeper program, and how it finds optimal yield allocations while mitigating risks for the funds. We will round off the section with the performance fee calculations.

4.1 System Overview

Meteora Dynamic Vaults allow users and integrated protocols to deposit and/or withdraw assets from the vault program any time. Deposited assets are distributed to various lending protocols like Solend & Tulip, with maximum allocation based on a combination of yield percentages and risk mitigation strategies around protocol audit, insurance coverage and open source status.

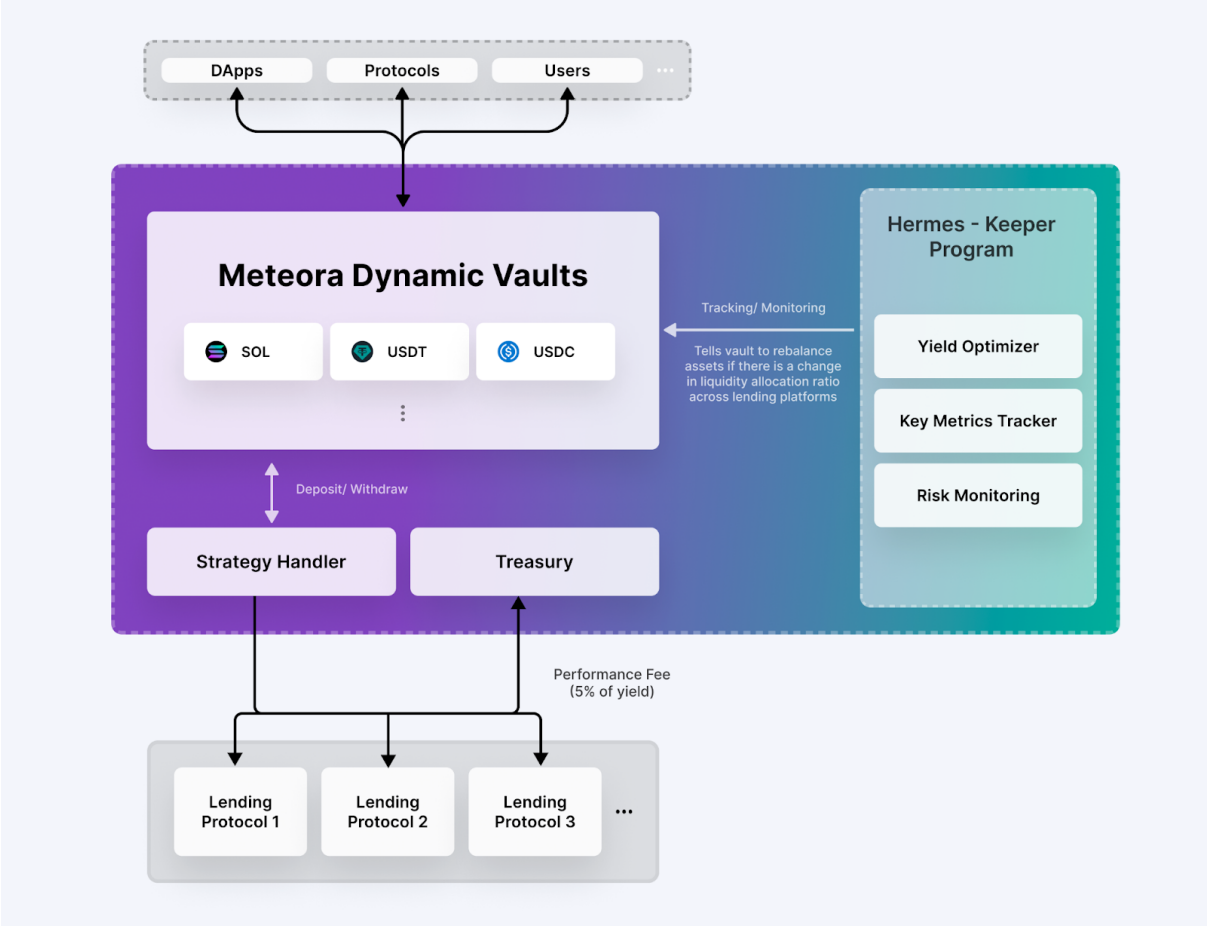


Fig 1: Meteora Dynamic Vaults overview

The system will consist of 3 main components:

1. Vault

Each Vault in the infra layer stores single token assets, e.g. USDC or SOL, and the majority of the assets will be allocated to various lending protocols to earn yield. The common tokens used in each connecting protocol, AMM or wallet will be stored in a single vault, e.g. USDC from AMM and the wallet will be held in the USDC vault. Users and protocols can deposit liquidity to each Vault directly through a simple interface.

2. Keeper - Hermes

We've created an off-chain keeper - Hermes to manage more complex logic and operations i.e. lending protocol monitoring and calculating optimal liquidity allocation across lending platforms etc. There are 3 main operations handled by Hermes:

Yield Optimizer - Hermes will calculate the liquidity allocation across the lending platforms that will generate the most optimal overall APY for the vault. The calculation will require key data from the various lending platforms i.e. Deposit APY, utilization rate, liquidity in pool etc. This process will repeat once every few minutes, and if there is a delta between the new allocation and the current one, a rebalance crank will be sent to the vault to trigger deposits and withdrawals to/from the lending platforms.

Key Metrics Tracker - As indicated above, the calculation of liquidity allocation requires various key data such as deposit APY and liquidity in pool from the lending platforms. The tracker will consistently monitor, track and store these information in the system for the use in calculations and for future references. These data are also exposed to potential integrators for them to display on their UIs or to support their own calculations or logics.

Risk Monitoring - Hermes also runs a risk monitoring service to track utilization rates and reserve levels of lending protocols to safeguard user assets, ready to withdraw liquidity when thresholds are reached. For example, if the utilization rate of a lending pool is above 80%, Hermes will send a transaction to the vault to trigger a full liquidity withdrawal from the pool. All deposits into the pool will also be stopped for 12 hours, giving us sufficient time to investigate and assess if we will continue deposits or stop them completely until further notice. Full details of the various mechanisms are discussed in section 4.3.3.

3. SDK Module (Integration APIs)

To make it easy for DApps and Protocols like AMMs and wallets to integrate with our Dynamic Yield Layer, we have created a straightforward SDK and are building up an entire library of pre-built modules and code samples for rapid app development and plug-and-play.

The liquidity in the protocols can be deposited into or withdrawn from the vaults directly via simple API calls. The vaults' yield can be distributed back to the LPs of the integrated protocols.

4.1.1 Definitions & Notation

This section describes and defines the notations used throughout this paper.

Name	Variable Name	Notation (if any)	Function
Total amount in vault	vault.total_amount	*t*	The total liquidity of the vault; equivalent to the sum of remaining tokens in the token vault and total liquidity deposited across all strategies.
Liquidity in token vault reserves	token_vault.amount	*a*	Actual amount of tokens in the token vault reserves.
Current liquidity in a strategy	strategy.current_liquidity	*c*	Total amount of liquidity deposit in a strategy
Total liquidity provider supply	total_lp_supply	-	Total amount of liquidity provided by LPs for a vault
Performance fee vault	fee_vault	*fee*	This vault holds the performance fee earned. Each time a rebalance crank is called, vault calculates performance fee and mints corresponding lp token amount to fee_vault. **fee_vault** is owned by **treasury address**
Total locked profit	Last_updated_locked_profit	-	Total locked profit where it is updated every time we do a rebalancing
Timestamp of last rebalancing run	Last_report	-	Store the timestamp of the last rebalancing run
Rate at which profit is unlocked	Locked_profit_degradation	-	Rate at which the profit will be unlocked in % per second e.g. 0.1% of locked_profits released per second.

Table 1: This table describes the state variables and their notations used in this paper

Operator describes the authorized wallet address that is only allowed to distribute funds into predefined protocols, it is not allowed to send liquidity to other places. The operator can only claim rewards and withdraw from predefined protocols to vault reserves, it is disallowed from withdrawing to external addresses.

Strategy refers to the lending protocols that the vaults will connect or deposit into.

Strategy Handler refers to the interface built to abstract all code that links the vault to the external lending platform to simplify handling.

Rebalance Crank happens when an operator deposits to or withdraws from a lending protocol. Rebalancing is run once every few minutes, and the yield optimizer will claim yield from the various lending protocols after each run.

Operation describes a generic action of interacting with the vaults, such as user depositing, user withdrawing, operator sending rebalance crank.

Utilization rates refer to the ratio of borrowed amount to deposited amount in a lending pool. A 100% utilization rate means that all deposited funds in a pool has been lent out.

4.2 Assertions

Meteora Dynamic Vaults distributes liquidity to various lending protocols like Solend, Port-Finance, Tulip, Raydium and Apricot.

To provide optimized yield while protecting the safety and liquidity of funds, any transfers in and out of the vaults must pass the following assertions:

1) The keeper can only send rebalance cranks to distribute funds to predefined protocols. It cannot claim principal funds for itself, or send liquidity to external wallets.

2) At any point in time, the *Total Amount in Vault* must equal or less than the summation of liquidity in token vault reserves and sum of liquidity distributed to the different lending protocols.

   vault.total_amount <= token_vault.amount + sum(liquidity_in_strategies)

3) The vaults preserve the principal and earn yield; the virtual price of the Liquidity Provider (LP) token is always increased for every single operation as long as the strategies earn yield. Virtual price of LP is calculated by the total amount in vault divided by total LP supply.

   Virtual price LP = vault.total_amount/total_lp_supply

   When strategies earn yield, it will add to the vault.total_amount value while keeping total_lp_supply the same, thus increasing the virtual price of the LP.

4.3 Hermes - Meteora Dynamic Vaults Keeper Program

In this section, we describe Hermes - our off-chain yield optimizer keeper program, diving into its algorithmic design for finding optimal yield allocations and the risk factors it helps to mitigate. Hermes contains the necessary program logic (monitoring and tracking of the lending pools) of the operator.

4.3.1 Algorithm to find optimal yield allocations

Annual Percentage Rate (APR) of lendings depends on a few factors: amount borrowed, amount deposited, interest rate model. The higher the amount borrowed, the higher the APR. When funds are deposited into a lending pool, borrowing interest will be shared amongst depositors, resulting in the decrease of deposit APR.

The algorithm to search for optimal yield allocations begins by breaking liquidity to small portions (lines1-2). For each portion, we simulate deposits in all lending platforms and compare the APR after the simulated deposit (lines 4-8). We will then pick the platform with the highest APR for this portion and update the deposit amount on the platform in the simulation (lines 10-12). We repeat this until 100% of total liquidity is deposited in the simulation to find the optimum allocation (lines 5-12). By doing so, we are able to find the most optimized yield allocations amongst the changing APR values with each deposit and withdrawal.

If the latest allocation differs from the last allocation by more than 0.1%, a rebalance crank is sent to withdraw or deposit into the lending protocols according to the latest allocation (lines 14-16).

```
# Off chain simulation
1:  portion ← x  # x is minimally 100, set by admin
2:  deposit_amount ← vault.total_amount / portion
3:  last_allocation[] ← current allocation of vault.total_amount in each lending platform
4:  allocation[] ← track allocation after simulation to each lending platform
5:  FOR each portion
6:      FOR each platform
7:          Simulate deposit_amount to platform
8:          APR[platform] ← APR of platform after simulated deposit
9:      ENDFOR
10:     highest_APR_platform ← Select platform with the highest APR in APR[platform]
11:     allocation[highest_APR_platform] ← deposit_amount + allocation[highest_APR_platform]
12:     Update deposit_amount to platform
13: ENDFOR

# On Chain Rebalance crank
14: IF diff(allocation[], last_allocation[]) > 0.1% THEN
15:     Send rebalance crank to allocate funds according to allocation[]
16: ENDIF
```

Figure 2: The optimal yield allocation algorithm

4.3.2 Rebalancing Crank Mechanism

This section describes the rebalance crank mechanism and calculations of how the vault total amount is updated after each rebalancing.

Hermes will claim yield from the various lending protocols after each rebalancing run. Rebalancing is run once every few minutes and the yield collected is included in the vault total amount as illustrated below.

We call state variables before the rebalance:

- vault.total_amount : t_1
- token_vault.amount : a_1
- strategy.current_liquidity : c_1

And the state variables after the rebalance:

- vault.total_amount : t_2
- token_vault.amount : a_2
- strategy.current_liquidity : c_2

Then the vault would know that the total accrued interest after rebalance is:

$$profit = (c_2 + a_2) - (c_1 + a_1)$$

The vault will then update the total amount:

$$t_2 = t_1 + profit$$

Or:

$$t_2 = t_1 + (c_2 + a_2) - (c_1 + a_1)$$

Rebalancing calculation illustration:

Event	a_1	c_1	a_2	c_2	t_1	t_2	$profit$
Before rebalancing	10	20	-	-	30	-	-
After rebalancing (10 token yield)	10	20	10	20 + 10 = 30	30	40	10

4.3.3 Risk factors & mitigation

Apart from executing the algorithm to find optimal yield allocations, our keeper program also has risks to consider before it can decide on the actual allocation. Risks are generally categorized into 2 types - Operation and lending risk.

Operation Risk: Risks that are related to source code such as when a partner protocol or team has a program update, or when lending platforms are not well audited. In minor cases, the source code changes break the integration, users are unable to perform any vault withdrawal or deposits. In major cases, the vault program or lending protocols may be exploited, losing the tokens in the vaults.

We implement a maximum allocation mechanism that the vault can deposit into each lending pool to mitigate this risk.

All lending protocols' maximum allocation starts at 100%. We will assess them across a set of criteria which includes the existence of audits, open-source code, insurance funds, main token pools, program multisig / verified & non updatable status as well as the length of integration with Meteora. This set of criteria will eventually be governed by the DAO.

For every criteria not met, we will reduce the maximum allocation allowed to the protocol according to this matrix:

Criteria	**Audit**	**Open-Source**	**Official Insurance Funds?**	**Main Pool**	**Existing integration > 1 month**	**Program multisig / or Verified & Non Updatable**
Maximum allocation reduction, if not present	20	30	20	10	10	20

Hermes is not allowed to withdraw funds from the lending protocols to external wallets. In the event if Hermes is hacked, the hackers will only be able to control the flow of funds to and fro between the vaults and lending protocols; the principals are still safe in either of them.

Lending Risk: This risk occurs when depositors are unable to withdraw their funds from the lending pools. This is caused when utilization rates of the reserves reach full capacity at 100% where borrowed amount equals deposited amount, or when the amount of reserves remaining in the lending pools are less than the vault deposits. When this happens, depositors are unable to withdraw funds on demand.

To avoid lending risks, we have developed the following mechanisms to protect principals:

- Stretch allocations in multiple lendings to diversify and manage risk across them
- Hermes consistently monitors utilization rates of each lending pool and is ready to withdraw funds whenever the threshold is exceeded. Current thresholds are set at 80% - allowing us to participate in popular lending pools with higher utilization rates while still leaving a buffer for Hermes to withdraw funds when required.
- Vaults always maintain a buffer in the lending reserve to allow Hermes buffer time to react for liquidity movements.

Section 5 Case Studies covers real life examples of the above mechanisms in play and how it helped protect user funds.

4.4 Sandwich Attack Prevention

The immediate release of profit generated to the LPs can result in opportunities for a Sandwich Attack as seen in the scenario below:

Event	Total amount in system	LP supply
Before rebalancing	100	100
Attacker deposits 100 tokens	200	200
After rebalancing (20 token yield)	220	200
Attacker withdraws 100 LP	110	100

If an attacker times their deposit before a rebalance and withdraws immediately once the rebalance runs, they will be able to maximise their profits through the Sandwich Attack.

In the example above, the attacker had deposited 100 tokens just before rebalancing, obtaining 50% of the LP tokens (100 LP tokens in this case). After rebalancing happens, there is a yield of 20 tokens and the total amount of tokens is 220 in the system.

Suppose we immediately release all 20 tokens of profits to LP and the attacker also withdraws his 50% share of the 100 LP tokens at this time, he would have taken away with him all 50% of the profits (10 tokens in this case) just for timing his deposits and withdrawals to the rebalancing runs.

Sandwich attacks are unfair to legitimate users and protocols who have deposited their assets to earn yield. It is important to ensure that the genuine liquidity providers are rewarded and the attackers do not profit at their expense.

Instead of distributing 100% of the yield generated by the lending platforms to the LPs immediately after rebalancing, the system will drip the yield to them across a pre-determined period of time. All earned profits from strategies are locked by default and will be subjected to a locked profit degradation rate where profit will be unlocked at a specified percentage per second.

4.4.1 Sandwich Attack Prevention Design

When a user adds or removes liquidity, instead of using vault.total_amount to calculate the total lp to mint or burn, we use the *get_unlock_amount f*unction to calculate the $unlocked\_amount$ value to use.

$$duration = current\_time - last\_report$$

$$locked\_fund\_ratio = 1 - duration\ x\ locked\_profit\_degradation$$

$$unlocked\_amount = vault.total\_amount - last\_updated\_locked\_profit\ x\ locked\_fund\_ratio$$

Using $unlocked\_amount$ to calculate how much a user gets during withdrawal can prevent attackers from doing a sandwich attack.

Potential attackers will not be able to get any of the yield gained if he were to withdraw his tokens immediately after rebalancing. He will also only gain a tiny fraction of the total if he withdraws within the next few minutes. This makes it unattractive to execute a sandwich attack since the yield earned is very small in this case.

4.6 Performance Fees

This section describes the performance fee we charge and its calculations. Fee is collected to the fee vault every time the operator sends a rebalance crank. The fee vault is controlled by the treasury and separate from the dynamic vaults. At this time of writing, we are currently charging 5% of profit as the performance fee.

We define variables as follows:

Before rebalance:

- vault.total_amount : t_1
- lp_mint.total_supply: p_1
- virtual_price (value of lp token): $v_1 = \frac{t_1}{p_1}$

After rebalance:

- vault.total_amount : t_2
- lp_mint.total_supply: p_1
- virtual_price: $v_2 = \frac{t_2}{p_1}$

We charge performance fee:

$$fee = 5\% * (t_2 - t_1) = 0.05 * (t_2 - t_1)$$

Virtual price after fee:

$$v_{21} = \frac{t_2 - fee}{p_1} \text{ (1)}$$

Vault does not send the token directly to the treasury token account (because the vault may not have enough liquidity), the vault will mint more lp tokens for **fee_vault**. Assuming vault mints more Δp lp tokens, then the new virtual price:

$$v_{22} = \frac{t_2}{p_1 + \Delta_p} \text{ (2)}$$

We ensure the virtual price in (1) and (2) are the same $v_{21} = v_{22}$, so

$$\frac{t_2 - fee}{p_1} = \frac{t_2}{p_1 + \Delta_p}$$

Then we can calculate how many lp tokens is minted more for each rebalance crank:

$$\Delta_p = \frac{p_1 * fee}{t_2 - fee}$$

Or

$$\Delta_p = \frac{p_1 * (t_2 - t_1)}{4 * t_2 + t_1}$$

5. Case Studies

In this section, we will examine 2 real life case studies that we experienced in our beta testing phase.

5.1 Mango Markets Exploit

On 12th October 2022, an exploiter was able to drain funds of over 100M USD from MANGO via an oracle price manipulation.[1]

Problem: Our dynamic vaults liquidity was affected by this exploit as MANGO is one of the lending platforms that we allocate assets to. The impact was limited as we were still in our Beta testing phase and imposed deposit limits to our users.

Impact: 900,000 USDC was locked up in MANGO as the funds were drained before we could withdraw them in time. MANGO paid us back through their own treasury and we were able to recover our locked funds.

Remarks: We were able to retrieve our funds through the existence of MANGO's insurance funds. Furthermore, this exploit also highlighted the areas in our safety mechanisms where we can fortify and improve them to further mitigate similar risks.

Details of the exploit from our POV here:
https://docs.google.com/document/d/1uL_3mEuszihwMkRqaJQTwnfzu1zJNSEkNg4LzjnC8Rg/edit#heading=h.1gcsvpgzo4pt

5.2 Solend.fi USDH Exploit

On 2nd Nov 2022, an exploiter inflated the price of the USDH stablecoin via an oracle attack on Saber and drained assets from Solend's isolated pools i.e. Stable, Coin98, and Kamino, resulting in $1.26M in bad debt.[2]

Problem: Although we did not support USDH assets, our vaults were exposed to this exploit as we have UXD assets in the Stable and Coin98 pools on Solend.

Impact: Hermes was monitoring the utilization rates of the pools and detected the high utilization of (>80%) of both Stable and Coin98 pools. A withdrawal request was immediately sent and all UXD assets were withdrawn back to our vaults safely before the pools were drained.

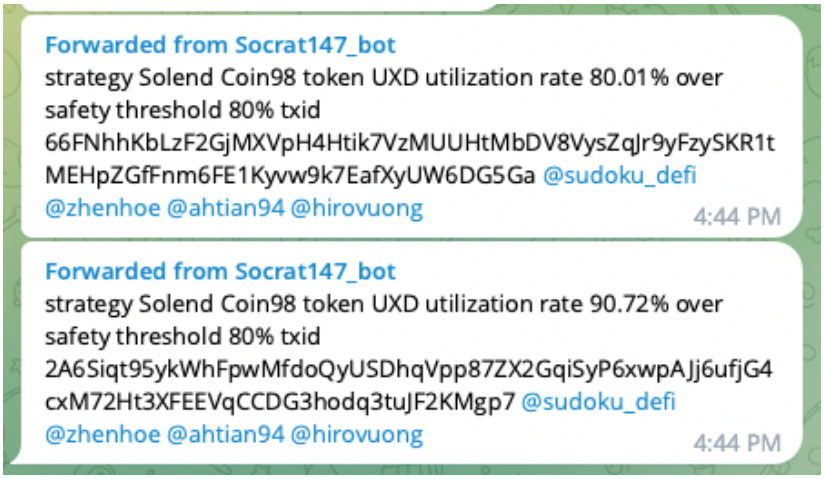


Fig 3: Keeper report of Solend utilization rate.

Remarks: This safety mechanism helped us avoid the lock up of user funds in Solend as we were able to withdraw 100% of UXD liquidity from Solend back to our vaults, proving its efficacy.

6. Dynamic Vaults as a Yield Layer in Solana

We believe Meteora Dynamic Vaults will be the yield layer for all of Solana as it allows any protocol, including wallets, treasuries and Automated Market Makers (AMMs) to build on top of this layer to generate more returns for their Liquidity Providers (LPs).

In this diagram, we can see the flow of liquidity into and out of the yield layer, through to the lending protocols and back to the vaults and users. Only one integration is needed to connect to the yield layer to access yield of the lending protocols.

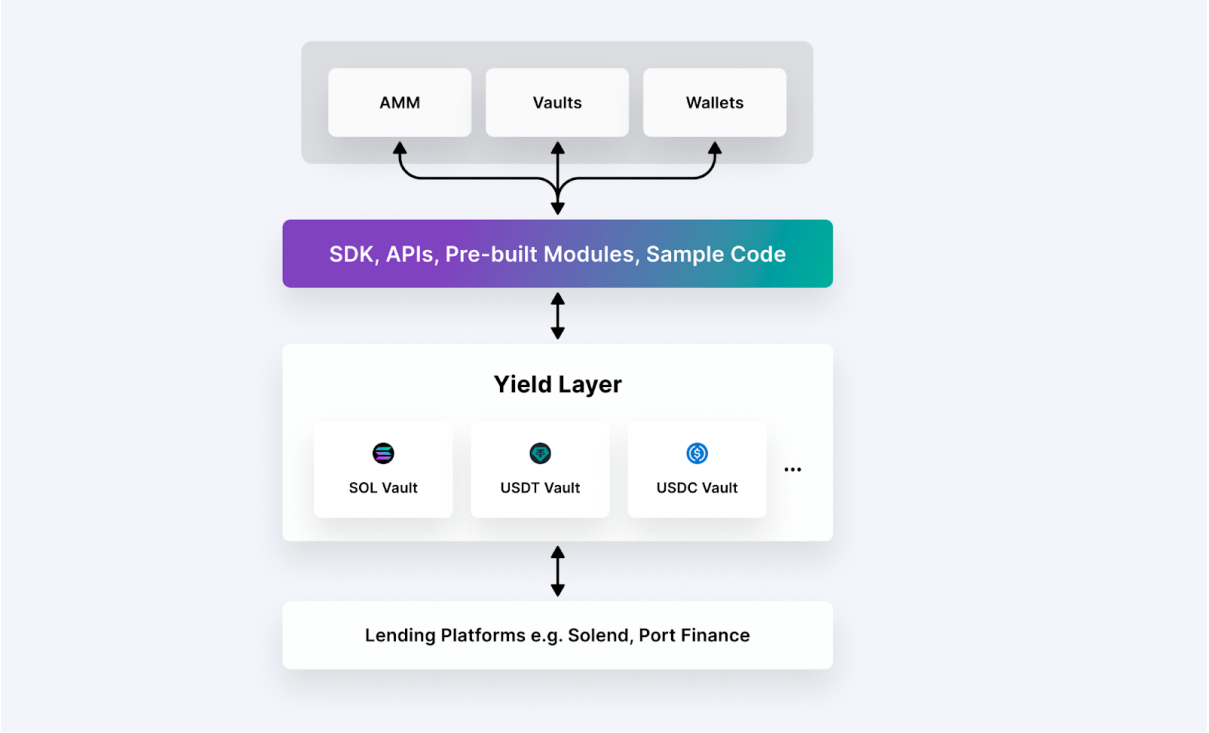


Fig 4: Overview of Dynamic Yield Layer

6.1 Increase utility of wallets, protocols & AMMs and DAOs

As mentioned earlier, the challenges of optimized yield, fund safety and liquidity access are not unique to users, they also extend to protocols that store liquidity in their system such as wallets, protocols & AMM pools and DAO treasuries. In this section, we will describe their problems faced and understand how to increase their utility.

Wallets contain assets for their users that they want to help them earn yield on. Protocols & AMMs have sizeable amounts of liquidity but scaling it is very inefficient. They rely heavily on liquidity mining to attract deposits. DAOs need to hold the treasury assets in an optimized and safe state.

Meteora Dynamic Vaults allows any protocol, including wallets, treasuries and AMMs, to easily build on top of them to generate more returns for their Liquidity Providers (LPs), overcoming the challenges of optimized yield, safety and liquidity access with one integration. These protocols can directly deposit and withdraw assets via the APIs in our SDK.

In addition, users who deposit into these protocols, integrated into our dynamic vaults, will be able to receive yield from the protocols, on top of the interest and LM rewards from the lending platforms. The added yield will significantly reduce LM as the primary driver of a protocol's liquidity maintenance and growth.

6.2 AMM Case Study

Currently, the vast majority of assets in AMMs are unutilized, as only a small portion of the assets is being constantly used for swaps. As a result, the yield generated is insufficient to attract LPs; continuous LM is needed to boost LP incentives instead, which is unsustainable.

However, suppose the AMMs are built on top of the yield layer. In that case, the liquidity of the pools will be deposited in the dynamic vaults and reallocated to various lending platforms to generate additional yield for the LPs. With the added yield, we will be able to make our pools highly capital efficient and reduce the reliance on LM to sustain or grow the liquidity of the pools.

Take an example of a USDC and USDT AMM stable pool being set up on top of the yield infrastructure.

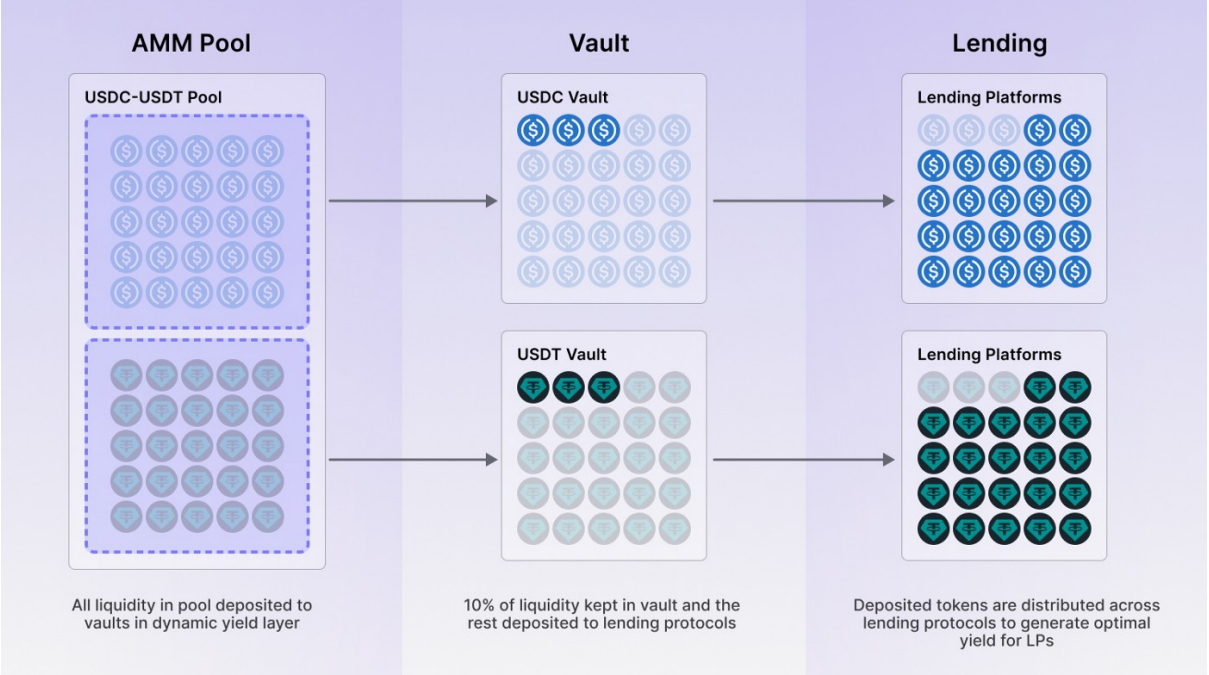
1. All the USDC/USDT tokens deposited in the AMM pool are immediately deposited into the USDC and USDT vaults in the infra layer.

2. The USDC and the USDT vaults will each keep 10% of the liquidity in the vault as reserves for the connecting AMMs to withdraw or swap tokens.

3. The remaining 90% of the tokens will be allocated to the various lending platforms integrated with the vaults. For instance, the USDC vault will distribute 90% of the USDC tokens in the vault across the pools in Port Finance, Solend and Mango to earn yield.

4. The yield optimizer will monitor and re-adjust the liquidity allocation ratio across Port Finance, Solend, Francium, Tulip and Apricot once every few minutes to obtain the optimal yield for the LPs.

The dormant tokens in the pools are now actively flowing and generating returns via the Yield Layer, making our pools extremely capital efficient.



We envision the dynamic vaults as the yield layer infrastructure of Solana that provides anyone with a robust yield layer to park and grow their assets.

7. Conclusion

In this paper, we shared about the key problems in managing assets and yield in a DeFI context, highlighting the 3 main challenges of optimized yield, funds security and liquidity access. We presented Meteora Dynamic Vaults - The Yield Layer for Solana, which aims to solve these challenges by seeking the most optimized yield for deposited assets through distributing liquidity into the various lending protocols and doing the heavy lifting of constant monitoring and fund rebalancing.

The features of Meteora Dynamic Vaults include Hermes - our keeper that optimizes yield and executes risk management strategies to deposit or withdraw from lending protocols, our reward handler to claim yield and rewards to distribute to the LPs and stakers,

Our design was made possible by leveraging on Solana's speed, composability and low transaction fees to extend event monitoring benefits for our system. This system design is extendable to any chains that can support event monitoring.

We envision Meteora Dynamic Vaults as the infrastructure that enables any protocol that stores liquidity (wallets, AMMs, DAOs etc.) to transform their idle funds into yield generating assets in a sustainable, safe and liquid manner.

References:

[1]: Mango Markets Exploit:
https://twitter.com/mangomarkets/status/1579979342423396352?lang=en
[2]: Solend Exploit:
https://twitter.com/solendprotocol/status/1587671511137398784?s=20&t=7H-bfCdfCUbI_W0pn5Gyww