Magen Protocol - Technical Resources

Introduction

Overview

Magen è un protocollo di assicurazione decentralizzata che trasforma la gestione del rischio in un mercato liquido e trasparente. Magen opera come un **Automated Market Maker (AMM)** dove ogni polizza assicurativa è contenuta all'interno di una **Insurance Pool** specifica.

Architecture

Smart Contracts List

MagenVault.sol

- `mint(uint amount)`: Riceve X USDC, conia X SI e X NO.
- `burn(uint amount)`: Riceve X SI e X NO, restituisce X USDC.
- `resolve(uint scale)`: Risolutore dell'evento. Fissa il valore di SI (es. 0.8) e NO (es. 0.2).
- `claim(uint amount, bool isSI)`: Brucia un singolo token e paga in base alla `scale`.

MagenRouter.sol (L'Interfaccia)

- `initialize(...)`: Gestisce il setup iniziale per il LP.
- `buySI(uint usdcIn)`: Permette all'utente di entrare solo con USDC.
- `sellSI(uint siIn)`: Permette all'utente di uscire in USDC.

SI / NO Tokens

- Token ERC-20 standard rappresentanti la protezione (SI) e il rendimento (NO).

Percorso Possibili

1. Creazione Liquidity Pool
   L'obiettivo è creare il mercato e impostare il prezzo (rischio) al 5%.
   1. **LP** chiama `Router.initialize(100.000 USDC)`.
   2. **Router** chiama `Vault.mint(100.000)`.
   3. **Vault** invia **100k SI** e **100k NO** al **Router**.
   4. **Router** crea la Pool su Uniswap e deposita **10.000 SI** e **526 NO**.
   5. **Router** invia i restanti **90.000 SI** e **99.474 NO** al wallet del **LP**.
   6. **Stato Finale:** Mercato pronto, prezzo SI circa 0.05 USDC.
2. Acquisto Protezione
   L'utente vuole assicurarsi per 1.000 USDC (Valore Totale di Copertura = 20.000 USDC)
   1. **User** chiama `Router.buySI(1.000 USDC)`.
   2. **Router** chiama `Vault.mint(1.000)` → Riceve **1k SI** e **1k NO**.
   3. **Router** va sulla Pool e scambia i **1.000 NO** → Ottiene (circa) **19.000 SI**.
   4. **Router** invia all'**User** un totale di **20.000 SI**.
3. Vendita Protezione
   L'utente vuole annullare la polizza prima della scadenza.
   1. **User** chiama `Router.sellSI(20.000 SI)`.
   2. **Router** va sulla Pool e vende una parte di SI (es. 1.000) per ottenere **1.000 NO**.
   3. **Router** ora ha una coppia bilanciata di **1.000 SI** e **1.000 NO**.
   4. **Router** chiama `Vault.burn(1.000)` $\rightarrow$ Riceve **1.000 USDC**.
   5. **Router** invia i **1.000 USDC** all'**User**.
4. Settlement e Claim
   L'oracolo decide l'esito dell'hack (es. 80% danno).
   1. **Oracolo** chiama `Vault.resolve(0.8)`. Il mercato è chiuso.
   2. **User** e **LP** ritirano la liquidità rimasta dalla Pool Uniswap (ora hanno solo token SI e NO "sciolti").
   3. **User** chiama `Vault.claim(20.000 SI)` $\rightarrow$ Riceve **16.000 USDC** (20.000×0.8).

4. **LP** chiama `Vault.claim(tutti i suoi SI e NO)` → Riceve il valore pesato (es. 80% per ogni SI e 20% per ogni NO).