Introduction

A smart contract security audit review of the init captial protocol was done by sparkware's auditor, 0xladboy233, with a focus on the security aspects of the application's implementation.

About Sparkware security

Sparkware security offers cutting edge and affordable smart contract auditing solution. The auditors get reputatoin and skill by consistently get top place in bug bounty and audit competition.

Our past audit prject including optimism and notional finance and graph protocol. Below is a list of comprehensive security review and research: https://github.com/JeffCX/Sparkware-audit-portfolio

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About **ProtocolName**

Init Captial is lending protocol in mantle network. the scope of audit focus on new swap helper integration with FusionX and agni dex and the new USDY oracle integration with ondo finance.

Security Assessment Summary

***review commit hash* - 5103e227424335590b391bf84c9b9cffdfd3da93**

Scope

The following smart contracts were in scope of the audit:

- `contracts/helper/swap_helper/AgniSwapHelper.sol`
- `contracts/helper/swap_helper/FusionXSwapHelper.sol`
- `contracts/oracle/usdy/UsdyOracleReader.sol`

Findings Summary

ID	Title	Severity
[L-01]	The USDY oracle can be paused by ondo admin	Low
[L-01]	Lack of price freshness check for USDY oracle	Low

Detailed Findings

L-01: The USDY oracle can be paused by ondo admin

the RWADynamicOracle can be paused by admin,

https://explorer.mantle.xyz/address/0xA96abbe61AfEdEB0D14a20440Ae7100D9aB4882f/contracts#address-tabs

```
function getPrice() public view whenNotPaused returns (uint256 price) {
  uint256 length = ranges.length;
  for (uint256 i = length; i > 0; --i) {
    Range storage range = ranges[i - 1];
    if (range.start <= block.timestamp) {
      if (range.end <= block.timestamp) {
        return derivePrice(range, range.end - 1);
      } else {
        return derivePrice(range, block.timestamp);
      }
    }
  }
}
```

It is mentioned that the protocol wants to support USDY as collateral only

but the code will still reply on the oracle to get accurate USDY price when user's USDY collateral is subject to liquidation if the oracle is paused, then user that wants to deposit USDY as collateral cannot to do,

but if the user's position is subject to liquidation and liquidator cannot complete liquidation because of the USDY Oracle pause, then the impact is more signficant and it can create bad debts over time.

Recommendation

it is recommended that the protocol implement a fallback oracle for USDY and do not hard to the USDY oracle address

https://github.com/init-capital/init-private-audit/blob/5103e227424335590b391bf84c9b9cffdfd3da93/contracts/oracle/usdy/UsdyOracleReader.sol#L12

```
    contract UsdyOracleReader is IUsdyOracleReader {
        // constants
        uint private constant ONE_E18 = 1e18;
        // immutables
        /// @inheritdoc IUsdyOracleReader
        address public RWA_DYNAMIC_ORACLE;

        address public owner;

        constructor(address _rwaDynamicOracle) {
            owner = msg.sender;
            RWA_DYNAMIC_ORACLE = _rwaDynamicOracle;
        }

        /// @inheritdoc IBaseOracle
        function getPrice_e36(address) external view returns (uint) {
            // IRWADynamicOracle returns usd price in 1e18
            return IRWADynamicOracle(RWA_DYNAMIC_ORACLE).getPrice() * ONE_E18;
        }

        function setOracleAddress(address _newOracleAddress) {
            require(msg.sender == owner, "invalid owner");
            RWA_DYNAMIC_ORACLE = RWA_DYNAMIC_ORACLE;
        }

    }
```

L-01: Lack of price freshness check for USDY oracle

https://docs.ondo.finance/developer-guides/mantle-integration-guidelines#usdy-oracle-rwadynamicrateoracle

> There is a functionality within the contract that if a range has elapsed and there is no subsequent range set, the oracle will return the maximum price of the previous range for all block.timestamp > Range.end

However, the code trust whatever value returned from the oracle price instead of validating if the price is fresh.

The latest price is not equal to fresh price in charge the oracle admin does not update the USDY price for a long period of time.

Recommendation

it is recommended to validate the lastest Range.end does not diverge too far away from the the current block.timestamp.

General recommendation:

R-01 adding thorough test to make sure the new swap helper works

it is highly recommended to add thorough test for the new swap helper implemented to make sure the code integrated correctly with the Trading hook.sol

R-01 ensure the swap path passed in is the most optimal

the swap helper is designed to help swap the token out before increase / reduce the position in the margin trading hook

https://github.com/init-capital/init-private-audit/blob/5103e227424335590b391bf84c9b9cffdfd3da93/contracts/hook/MarginTradingHook.sol#L95

it is highly recommended that the most optimal swap path is always passed via the parameter data