// HALBORN

Jarvis Exchange

Synthereum PerpetualPoolParty Smart Contract Security Audit

Prepared by: **Halborn**
Date of Engagement: December 27, 2020
Visit: Halborn.com

Document Revision History 3
Contacts 3
1 Executive Summary 4
1.1 Introduction 4
1.2 Test Approach and Methodology 5
1.3 SCOPE 5
2 Assessment Summary And Findings Overview 6
3 Findings & Technical Details 7
3.1 Missing Reentrancy Protection - Low 8
Description 8
Recommendation 8
3.2 Use Of Block.Timestamp - Low 8
Description 8
Recommendation 9
3.3 Ignore Return Values - Informational 9
Description 9
Recommendation 10-12
3.4 Tautology Expressions - Informational 12
Description 12
Recommendation 13
3.5 Static Analysis - Low 13
Description 13
Results 14-17
3.6 Erc Conformal Checker - Informational 17
Description 17
Recommendation 17

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/27/2020	Nishit Majithia
0.2	Document Edits	11/30/2020	Nishit Majithia
1.0	Final Version	12/01/2020	Nishit Majithia

CONTACTS

CONTACT	COMPANY	EMAIL
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Nishit Majithia	Halborn	nishit.majithia@halborn.com

1.1 INTRODUCTION

JARVIS engaged Halborn to conduct a security assessment on all of their smart contracts related to UMA fork functioning perpetual-poolparty. The security assessment was scoped to PerpetualLiquidatablePoolParty.sol, PerpetualLiquidatablePoolPartyLib.sol, PerpetualPoolParty.sol, PerpetualPoolPartyLib.sol, PerpetualPositionManagerPoolParty.sol, PerpetualPositionManagerPoolPartyLib.sol, PerpetutalPoolPartyCreator.sol and an audit of the security risk and implications regarding the changes introduced by the development team at JARVIS prior to its production release shortly following the assessments deadline.

Overall, the smart contracts code is well documented, follows a high-quality software development standard, contain many utilities and automation scripts to support continuous deployment / testing / integration, and does NOT contain any obvious exploitation vectors that Halborn was able to leverage within the timeframe of testing allotted.

Though the outcome of this security audit is satisfactory; due to time and resource constraints, only testing and verification of essential properties were performed to achieve objectives and deliverables set in the scope. It is important to remark the use of the best practices for secure smart contract development. Halborn recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit.

While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart Contract manual code read and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual Assessment of use and safety for the critical solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Scanning of solidity files for vulnerabilities, security hotspots, or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Truffle, Ganache, Infura) Smart Contract Fuzzing and dynamic state exploitation (Echidna) Symbolic Execution / EVM bytecode security assessment (limited time)

1.2 SCOPE

IN-SCOPE:

Perpetual-PoolParty smart contracts including

- PerpetualLiquidatablePoolParty.sol
- PerpetualLiquidatablePoolPartyLib.sol
- PerpetualPoolParty.sol
- PerpetualPoolPartyLib.sol
- PerpetualPositionManagerPoolParty.sol

- PerpetualPositionManagerPoolPartyLib.sol
- PerpetutalPoolPartyCreator.sol.

Specific commit of contract: commit 15f04ed70f68918a1029090994fed7a0ba72287a

OUT-OF-SCOPE:
External contracts, External Oracles, other smart contracts in the repository or imported by Perpetual-PoolParty smart contracts and rest of the UMA contracts, economic attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	0	2

SECURITY ANALYSIS	RISK LEVEL
ADDRESS CHECK MISSING	Low
DIVIDE BEFORE MULTIPLE	Informational
IGNORE RETURN VALUES	Informational
STATIC ANALYSIS	Low

FINDINGS & TECH DETAILS

3.1 ADDRESS CHECK MISSING - LOW

Description

Address validation check is missing in PerpetualPoolPartyCreator.sol contract's constructor() for _tokenFactoryAddress. The value of _tokenFactoryAddress which is passed during initialization is being directly assigned to the the tokenFactoryAddress which can be zero or empty or invalid.

Code Location

PerpetutalPoolPartyCreator.sol: Line #63 (https://gitlab.com/jarvis-network/apps/exchange/UMAprotocol/-/blob/15f04ed70f68918a1029090994fed7a0ba72287a/packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#L63)

```
58    constructor(
59        address _finderAddress,
60        address _tokenFactoryAddress,
61        address _timerAddress
62    ) public ContractCreator(_finderAddress) Testable(_timerAddress) nonReentrant() {
63        tokenFactoryAddress = _tokenFactoryAddress;
64    }
```

Recommendation: Check that the address is not zero or invalid before storing it into any state variable.

3.2 DIVIDE BEFORE MULTIPLE - INFORMATIONAL

Description:

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. In this audit, there are multiple instances found where division is being performed before multiplication operation in PerpetualPositionManagerPoolPartyLib.sol and PerpetualLiquiditablePoolPartyLib.sol.

Recommendation:

Consider doing multiplication operation before division to prevail precision in the values in non floating data type.

Code Location:

PerpetualPositionManagerPoolPartyLib.sol: Line #216-217 (https://gitlab.com/jarvis-network/apps/exchange/UMAprotocol/-/blob/15f04ed70f68918a1029090994fed7a0ba72287a/packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#L216)

```
215
216         FixedPoint.Unsigned memory fractionRedeemed = numTokens.div(positionData.tokensOutstanding);
217         FixedPoint.Unsigned memory collateralRedeemed =
218             fractionRedeemed.mul(
219                 positionData.rawCollateral.getFeeAdjustedCollateral(feePayerData.cumulativeFeeMultiplier)
220             );
221
```

PerpetualLiquidatablePoolPartyLib.sol: Line #415-419, 423, 427 (https://gitlab.com/jarvis-network/apps/exchange/UMAprotocol/-/blob/15f04ed70f68918a1029090994fed7a0ba72287a/packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#L415)

```
414         {
415             FixedPoint.Unsigned memory ratio =
416                 liquidationCollateralParams.tokensLiquidated.div(positionToLiquidate.tokensOutstanding);
417
418             // The actual amount of collateral that gets moved to the liquidation.
419             lockedCollateral = liquidationCollateralParams.startCollateral.mul(ratio);
420
421             // For purposes of disputes, it's actually this liquidatedCollateral value that's used. This value is net of
422             // withdrawal requests.
423             liquidatedCollateral = liquidationCollateralParams.startCollateralNetOfWithdrawal.mul(ratio);
424
425             // Part of the withdrawal request is also removed. Ideally:
426             // liquidatedCollateral + withdrawalAmountToRemove = lockedCollateral.
427             FixedPoint.Unsigned memory withdrawalAmountToRemove =
428                 positionToLiquidate.withdrawalRequestAmount.mul(ratio);
429
430             positionToLiquidate.reduceSponsorPosition(
431                 globalPositionData,
432                 positionManagerData,
433                 liquidationCollateralParams.tokensLiquidated,
434                 lockedCollateral,
435                 withdrawalAmountToRemove,
436                 feePayerData,
437                 liquidationCollateralParams.sponsor
438             );
439         }
```

3.3 IGNORE RETURN VALUES - INFORMATIONAL

Description:

The return value of an external call is not stored in a local or state variable. In contracts PerpetualPositionManagerPoolPartyLib.sol and PerpetualLiquidatablePoolPartyLib.sol. there are few instances where external methods are being called and return values(FixedPoint.Unsigned memory) are being ignored.

Recommendation:
Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Code Locatioin:
PerpetualPositionManagerPoolPartyLib.sol: Line #56, #393, #398, #437, #455, #469 (https://gitlab.com/jarvis-network/apps/exchange/UMAprotocol/-/blob/15f04ed70f68918a1029090994fed7a0ba72287a/packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol)

PerpetualLiquidatablePoolPartyLib.sol: Line #223, #442 (https://gitlab.com/jarvis-network/apps/exchange/UMAprotocol/-/blob/15f04ed70f68918a1029090994fed7a0ba72287a/packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol)

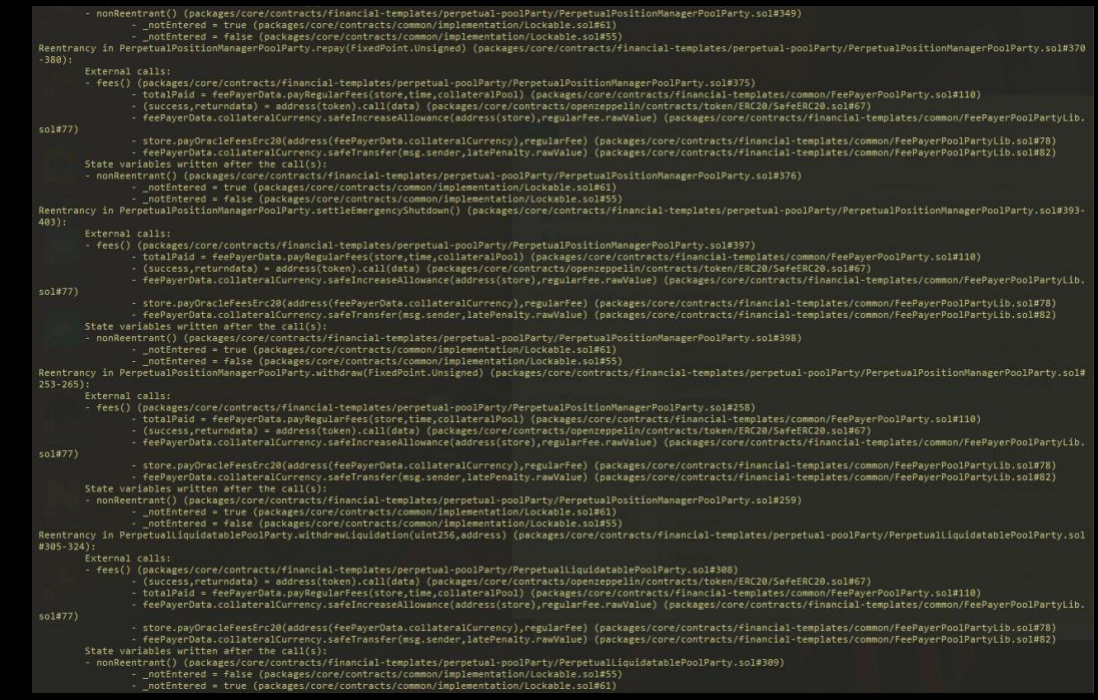
3.4 STATIC ANALYSIS - LOW

Description:
Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire codebase.

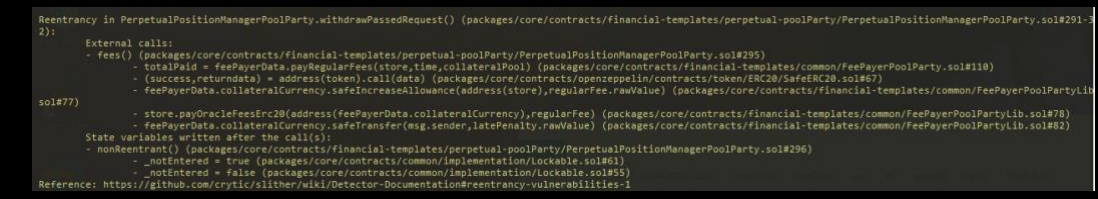
```
INFO:Detectors:
PerpetualLiquidatablePoolPartyLib (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#11-508) contract sets array length with a user-controlled value:
        - liquidations.push(PerpetualLiquidatablePoolParty.LiquidationData(params.sponsor,msg.sender,PerpetualLiquidatablePoolParty.Status.PreDispute,params.actualTime,returnValues.tokensLiquidated,returnValues.lockedCollateral,returnValues.liquidatedCollateral,FixedPoint.fromUnscaledUint(1).convertToRawCollateral(feePayerData.cumulativeFeeMultiplier),address(0),FixedPoint.fromUnscaledUint(0),returnValues.finalFeeBond)) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#154-170)
Registry (packages/core/contracts/oracle/implementation/Registry.sol#15-201) contract sets array length with a user-controlled value:
        - partyMap[party].contracts.push(contractAddress) (packages/core/contracts/oracle/implementation/Registry.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
INFO:Detectors:
PerpetualPositionManagerPoolParty.positions (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#122) is a public mapping with nested variables
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-mappings-with-nested-variables
```

```
INFO:Detectors:
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,PerpetualLiquidatablePoolParty.LiquidatableData,FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#405-446) performs a multiplication on the result of a division:
        -ratio = liquidationCollateralParams.tokensLiquidated.div(positionToLiquidate.tokensOutstanding) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#415-416)
        -lockedCollateral = liquidationCollateralParams.startCollateral.mul(ratio) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#419)
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,PerpetualLiquidatablePoolParty.LiquidatableData,FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#405-446) performs a multiplication on the result of a division:
        -ratio = liquidationCollateralParams.tokensLiquidated.div(positionToLiquidate.tokensOutstanding) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#415-416)
        -liquidatedCollateral = liquidationCollateralParams.startCollateralNetOfWithdrawal.mul(ratio) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#423)
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,PerpetualLiquidatablePoolParty.LiquidatableData,FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#405-446) performs a multiplication on the result of a division:
        -ratio = liquidationCollateralParams.tokensLiquidated.div(positionToLiquidate.tokensOutstanding) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#415-416)
        -withdrawalAmountToRemove = positionToLiquidate.withdrawalRequestAmount.mul(ratio) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#427-428)
PerpetualPositionManagerPoolPartyLib.redeeem(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,FixedPoint.Unsigned,FeePayerPoolParty.FeePayerData,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#206-251) performs a multiplication on the result of a division:
        -fractionRedeemed = numTokens.div(positionData.tokensOutstanding) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#216)
        -collateralRedeemed = fractionRedeemed.mul(positionData.rawCollateral.getFeeAdjustedCollateral(feePayerData.cumulativeFeeMultiplier)) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#217-220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PerpetualPositionManagerPoolParty._notEmergencyShutdown() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#683-685) uses a dangerous strict equality:
        - require(bool,string)(positionManagerData.emergencyShutdownTimestamp == 0,Contract emergency shutdown) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
INFO:Detectors:
Reentrancy in PerpetualPositionManagerPoolParty.create(FixedPoint.Unsigned,FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#322-332):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#326)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#327)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualLiquidatablePoolParty.createLiquidation(address,FixedPoint.Unsigned,FixedPoint.Unsigned,FixedPoint.Unsigned,uint256) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#215-265):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#223)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#225)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
Reentrancy in PerpetualPositionManagerPoolParty.depositTo(address,FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#222-233):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#227)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#228)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualLiquidatablePoolParty.dispute(uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#277-293):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#280)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#281)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
Reentrancy in PerpetualPositionManagerPoolParty.redeem(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#333-361):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#348)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
```



```
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#349)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualPositionManagerPoolParty.repay(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#370
-380):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#375)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.
sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#376)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualPositionManagerPoolParty.settleEmergencyShutdown() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#393-
403):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#397)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.
sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#398)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualPositionManagerPoolParty.withdraw(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#
253-265):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#258)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.
sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#259)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reentrancy in PerpetualLiquidatablePoolParty.withdrawLiquidation(uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol
#305-324):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#308)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.
sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#309)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
```



```
Reentrancy in PerpetualPositionManagerPoolParty.withdrawPassedRequest() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#291-3
2):
        External calls:
        - fees() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#295)
                - totalPaid = feePayerData.payRegularFees(store,time,collateralPool) (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#110)
                - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
                - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib
sol#77)
                - store.payOracleFeesErc20(address(feePayerData.collateralCurrency),regularFee) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#78)
                - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#82)
        State variables written after the call(s):
        - nonReentrant() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#296)
                - _notEntered = true (packages/core/contracts/common/implementation/Lockable.sol#61)
                - _notEntered = false (packages/core/contracts/common/implementation/Lockable.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
PerpetualLiquidatablePoolPartyLib.withdrawLiquidation(PerpetualLiquidatablePoolParty.LiquidationData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualPositionManagerPoolPar
y.PositionManagerData,FeePayerPoolParty.FeePayerData,uint256,address).settleParams (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLi
.sol#263) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PerpetualLiquidatablePoolPartyLib.dispute(PerpetualLiquidatablePoolParty.LiquidationData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualPositionManagerPoolParty.PositionM
nagerData,FeePayerPoolParty.FeePayerData,uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#210-250) ignores re
urn value by liquidatableData.rawLiquidationCollateral.addCollateral(disputeBondAmount,feePayerData.cumulativeFeeMultiplier) (packages/core/contracts/financial-templates/perpetual-
oolParty/PerpetualLiquidatablePoolPartyLib.sol#223-226)
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPo
lParty.PositionManagerData,PerpetualLiquidatablePoolParty.LiquidatableData,FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (packages/c
re/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#405-446) ignores return value by liquidatableData.rawLiquidationCollateral.addCollateral(
ockedCollateral.add(liquidationCollateralParams.finalFeeBond),feePayerData.cumulativeFeeMultiplier) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquid
tablePoolPartyLib.sol#442-445)
PerpetualPositionManagerPoolPartyLib.depositTo(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,FixedPoint.Unsigned,FeePayerPoolP
rty.FeePayerData,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#48-63) ignores return value by positionData._inc
ementCollateralBalances(globalPositionData,collateralAmount,feePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#
7)
PerpetualPositionManagerPoolPartyLib.reduceSponsorPosition(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionMana
erPoolParty.PositionManagerData,FixedPoint.Unsigned,FixedPoint.Unsigned,FixedPoint.Unsigned,FeePayerPoolParty.FeePayerData,address) (packages/core/contracts/financial-templates/per
etual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#376-412) ignores return value by positionData._deleteSponsorPosition(globalPositionData,feePayerData,sponsor) (packages/cor
/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#393)
PerpetualPositionManagerPoolPartyLib.reduceSponsorPosition(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionMana
erPoolParty.PositionManagerData,FixedPoint.Unsigned,FixedPoint.Unsigned,FixedPoint.Unsigned,FeePayerPoolParty.FeePayerData,address) (packages/core/contracts/financial-templates/per
etual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#376-412) ignores return value by positionData._decrementCollateralBalances(globalPositionData,collateralToRemove,feePayerDa
a) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#398)
PerpetualPositionManagerPoolPartyLib._incrementCollateralBalances(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,FixedPoint.Uns
gned,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#431-443) ignores return value by posi
ionData.rawCollateral.addCollateral(collateralAmount,feePayerData.cumulativeFeeMultiplier) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManager
oolPartyLib.sol#437)
PerpetualPositionManagerPoolPartyLib._decrementCollateralBalances(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,FixedPoint.Uns
gned,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#449-461) ignores return value by posi
ionData.rawCollateral.removeCollateral(collateralAmount,feePayerData.cumulativeFeeMultiplier) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionMana
erPoolPartyLib.sol#455)
PerpetualPositionManagerPoolPartyLib._decrementCollateralBalancesCheckGCR(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,Fixed
int.Unsigned,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#463-476) ignores return value
by positionData.rawCollateral.removeCollateral(collateralAmount,feePayerData.cumulativeFeeMultiplier) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPosi
ionManagerPoolPartyLib.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

FINDINGS & TECH DETAILS

```
INFO:Detectors:
PerpetualPositionManagerPoolParty.constructor(PerpetualPositionManagerPoolParty.PositionManagerParams,PerpetualPositionManagerPoolParty.Roles)._roles (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#183) shadows:
        - AccessControl._roles (packages/core/contracts/openzeppelin/contracts/access/AccessControl.sol#47) (state variable)
ERC20.constructor(string,string).name (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#55) shadows:
        - ERC20.name() (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#64-66) (function)
ERC20.constructor(string,string).symbol (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#55) shadows:
        - ERC20.symbol() (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#72-74) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
PerpetualPoolPartyCreator.constructor(address,address,address)._tokenFactoryAddress (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#60) lacks a zero-check on :
                - tokenFactoryAddress = _tokenFactoryAddress (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PerpetualPositionManagerPoolPartyLib._deleteSponsorPosition(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,FeePayerPoolParty.FeePayerData,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#526-556):
        External calls:
        - PerpetualPositionManagerPoolParty(address(this)).deleteSponsorPosition(sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#545)
        Event emitted after the call(s):
        - EndedSponsorPosition(sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#547)
Reentrancy in PerpetualLiquidatablePoolPartyLib.createLiquidation(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualLiquidatablePoolParty.LiquidationData[],PerpetualLiquidatablePoolPartyLib.CreateLiquidationParams,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#95-208):
        External calls:
        - (returnValues.lockedCollateral,returnValues.liquidatedCollateral) = liquidateCollateral(positionToLiquidate,globalPositionData,positionManagerData,liquidatableData,feePayerData,liquidationCollateral) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#141-148)
                - positionData._deleteSponsorPosition(globalPositionData,feePayerData,sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#393)
                - PerpetualPositionManagerPoolParty(address(this)).deleteSponsorPosition(sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#545)
                - positionToLiquidate.reduceSponsorPosition(globalPositionData,positionManagerData,liquidationCollateralParams.tokensLiquidated,lockedCollateral,withdrawalAmountToRemove,feePayerData,liquidationCollateralParams.sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#430-438)
        Event emitted after the call(s):
        - LiquidationCreated(params.sponsor,msg.sender,returnValues.liquidationId,returnValues.tokensLiquidated.rawValue,returnValues.lockedCollateral.rawValue,returnValues.liquidatedCollateral.rawValue,params.actualTime) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#192-200)
Reentrancy in PerpetualPoolPartyCreator.createPerpetual(PerpetualPoolPartyCreator.Params) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#71-106):
        External calls:
        - tokenCurrency = tf.createToken(params.syntheticName,params.syntheticSymbol,18) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#80)
        - tokenCurrency.addAdminAndMinterAndBurner(derivative) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#84)
        - tokenCurrency.renounceAdmin() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#85)
        - _registerContract(new address[](0),address(derivative)) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#101)
                - registry.registerContract(parties,contractToRegister) (packages/core/contracts/oracle/implementation/ContractCreator.sol#29)
        Event emitted after the call(s):
        - CreatedPerpetual(address(derivative),msg.sender) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#103)
```

```
Reentrancy in PerpetualLiquidatablePoolPartyLib.dispute(PerpetualLiquidatablePoolParty.LiquidationData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualPositionManagerPoolParty.PositionManagerData,FeePayerPoolParty.FeePayerData,uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#210-250):
        External calls:
        - positionManagerData.requestOraclePrice(disputedLiquidation.liquidationTime,feePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#233)
        Event emitted after the call(s):
        - LiquidationDisputed(sponsor,disputedLiquidation.liquidator,msg.sender,liquidationId,disputeBondAmount.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#235-241)
Reentrancy in PerpetualPositionManagerPoolParty.emergencyShutdown() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#415-419):
        External calls:
        - positionManagerData.requestOraclePrice(positionManagerData.emergencyShutdownTimestamp,feePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#417)
        Event emitted after the call(s):
        - EmergencyShutdown(msg.sender,positionManagerData.emergencyShutdownTimestamp) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#418)
Reentrancy in FeePayerPoolPartyLib.payFinalFees(FeePayerPoolParty.FeePayerData,StoreInterface,address,FixedPoint.Unsigned) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#91-108):
        External calls:
        - feePayerData.collateralCurrency.safeTransferFrom(payer,address(this),amount.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#102)
        Event emitted after the call(s):
        - FinalFeesPaid(amount.rawValue) (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#104)
Reentrancy in PerpetualPositionManagerPoolPartyLib.redeeem(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,FixedPoint.Unsigned,FeePayerPoolParty.FeePayerData,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#206-251):
        External calls:
        - amountWithdrawn = positionData._deleteSponsorPosition(globalPositionData,feePayerData,sponsor) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#224)
        Event emitted after the call(s):
        - Redeem(msg.sender,amountWithdrawn.rawValue,numTokens.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#245)
Reentrancy in PerpetualPositionManagerPoolPartyLib.settleEmergencyShutdown(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#279-346):
        External calls:
        - PerpetualPositionManagerPoolParty(address(this)).deleteSponsorPosition(msg.sender) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#319)
        Event emitted after the call(s):
        - EndedSponsorPosition(msg.sender) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#320)
        - SettleEmergencyShutdown(msg.sender,amountWithdrawn.rawValue,tokensToRedeem.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#340)
```

```
Reentrancy in PerpetualLiquidatablePoolPartyLib.withdrawLiquidation(PerpetualLiquidatablePoolParty.LiquidationData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualPositionManagerPoolParty.PositionManagerData,FeePayerPoolParty.FeePayerData,uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#252-371):
        External calls:
        - feePayerData.collateralCurrency.safeTransfer(liquidation.disputer,rewards.paidToDisputer.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#324)
        - feePayerData.collateralCurrency.safeTransfer(liquidation.liquidator,rewards.paidToLiquidator.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#325)
        - feePayerData.collateralCurrency.safeTransfer(liquidation.sponsor,rewards.paidToSponsor.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#326)
        - feePayerData.collateralCurrency.safeTransfer(liquidation.liquidator,rewards.paidToLiquidator.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#341)
        - feePayerData.collateralCurrency.safeTransfer(liquidation.liquidator,rewards.paidToLiquidator.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#355)
        Event emitted after the call(s):
        - LiquidationWithdrawn(msg.sender,rewards.paidToLiquidator.rawValue,rewards.paidToDisputer.rawValue,rewards.paidToSponsor.rawValue,liquidation.state,settleParams.settlementPrice.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#358-365)
Reentrancy in PerpetualPositionManagerPoolPartyLib.withdrawPassedRequest(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,uint256,FeePayerPoolParty.FeePayerData) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#112-147):
        External calls:
        - feePayerData.collateralCurrency.safeTransfer(msg.sender,amountWithdrawn.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#144)
        Event emitted after the call(s):
        - RequestWithdrawalExecuted(msg.sender,amountWithdrawn.rawValue) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

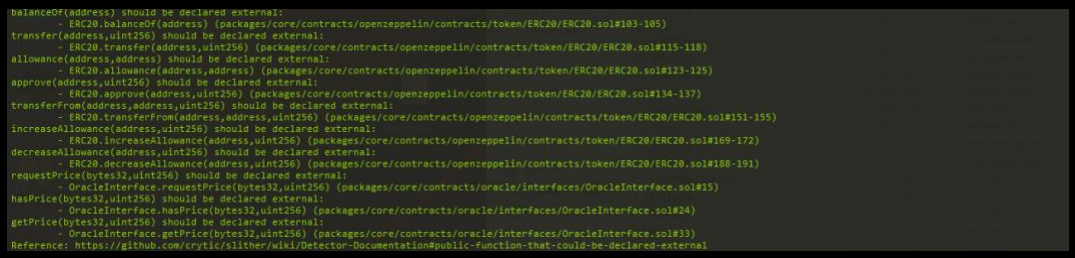
```
INFO:Detectors:
PerpetualLiquidatablePoolParty._disputable(uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#369-375) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((getCurrentTime() < _getLiquidationExpiry(liquidation)) && (liquidation.state == Status.PreDispute),Liquidation not disputable) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#371-374)
PerpetualLiquidatablePoolParty._withdrawable(uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#377-387) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((state > Status.PreDispute) || ((_getLiquidationExpiry(liquidation) <= getCurrentTime()) && (state == Status.PreDispute)),Liquidation not withdrawable) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#382-386)
PerpetualPositionManagerPoolParty._onlyCollateralizedPosition(address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#673-680) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(positions[sponsor].rawCollateral.getFeeAdjustedCollateral(feePayerData.cumulativeFeeMultiplier).isGreaterThan(0),Position has no collateral) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#674-680)
PerpetualPositionManagerPoolParty._notEmergencyShutdown() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#683-685) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(positionManagerData.emergencyShutdownTimestamp == 0,Contract emergency shutdown) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#684)
PerpetualPositionManagerPoolParty._isEmergencyShutdown() (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#687-689) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(positionManagerData.emergencyShutdownTimestamp != 0,Contract not emergency shutdown) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#688)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (packages/core/contracts/openzeppelin/contracts/utils/Address.sol#24-33) uses assembly
        - INLINE ASM (packages/core/contracts/openzeppelin/contracts/utils/Address.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

FINDINGS & TECH DETAILS

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['^0.6.0', '^0.6.2']
        - ^0.6.0 (packages/core/contracts/common/implementation/AddressWhitelist.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/FixedPoint.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/Lockable.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/MintableBurnableERC20.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/MultiRole.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/Testable.sol#2)
        - ^0.6.0 (packages/core/contracts/common/implementation/Timer.sol#2)
        - ^0.6.0 (packages/core/contracts/common/interfaces/IERC20Standard.sol#2)
        - ^0.6.0 (packages/core/contracts/common/interfaces/MintableBurnableIERC20.sol#2)
        - ^0.6.0 (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/common/FeePayerPoolPartyLib.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/common/MintableBurnableSyntheticToken.sol#2)
        - ^0.6.0 (packages/core/contracts/financial-templates/common/MintableBurnableTokenFactory.sol#2)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolParty.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolParty.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolPartyLib.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolPartyLib.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#3)
        - ^0.6.0 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#2)
        - ABIEncoderV2 (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol#3)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/GSN/Context.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/access/AccessControl.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/access/Ownable.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/math/SafeMath.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/math/SignedSafeMath.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/token/ERC20/IERC20.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#1)
        - ^0.6.2 (packages/core/contracts/openzeppelin/contracts/utils/Address.sol#1)
        - ^0.6.0 (packages/core/contracts/openzeppelin/contracts/utils/EnumerableSet.sol#1)
        - ^0.6.0 (packages/core/contracts/oracle/implementation/Constants.sol#2)
        - ^0.6.0 (packages/core/contracts/oracle/implementation/ContractCreator.sol#2)
        - ^0.6.0 (packages/core/contracts/oracle/implementation/Registry.sol#2)
        - ABIEncoderV2 (packages/core/contracts/oracle/implementation/Registry.sol#3)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/AdministrateeInterface.sol#2)
        - ABIEncoderV2 (packages/core/contracts/oracle/interfaces/AdministrateeInterface.sol#3)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/FinderInterface.sol#2)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/IdentifierWhitelistInterface.sol#2)
        - ABIEncoderV2 (packages/core/contracts/oracle/interfaces/IdentifierWhitelistInterface.sol#4)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/OracleInterface.sol#2)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/RegistryInterface.sol#2)
        - ABIEncoderV2 (packages/core/contracts/oracle/interfaces/RegistryInterface.sol#4)
        - ^0.6.0 (packages/core/contracts/oracle/interfaces/StoreInterface.sol#2)
        - ABIEncoderV2 (packages/core/contracts/oracle/interfaces/StoreInterface.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
INFO:Detectors:
Low level call in SafeERC20._callOptionalReturn(IERC20,bytes) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#55-74):
        - (success,returndata) = address(token).call(data) (packages/core/contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#67)
Low level call in Address.sendValue(address,uint256) (packages/core/contracts/openzeppelin/contracts/utils/Address.sol#51-57):
        - (success) = recipient.call{value: amount}() (packages/core/contracts/openzeppelin/contracts/utils/Address.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Constant OracleInterfaces.Oracle (packages/core/contracts/oracle/implementation/Constants.sol#8) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.IdentifierWhitelist (packages/core/contracts/oracle/implementation/Constants.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.Store (packages/core/contracts/oracle/implementation/Constants.sol#10) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.FinancialContractsAdmin (packages/core/contracts/oracle/implementation/Constants.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.Registry (packages/core/contracts/oracle/implementation/Constants.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.CollateralWhitelist (packages/core/contracts/oracle/implementation/Constants.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant OracleInterfaces.OptimisticOracle (packages/core/contracts/oracle/implementation/Constants.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (packages/core/contracts/openzeppelin/contracts/GSN/Context.sol#23)" inContext (packages/core/contracts/openzeppelin/contracts/GSN/Context.sol#13-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable PerpetualLiquidatablePoolParty.createLiquidation(address,FixedPoint.Unsigned,FixedPoint.Unsigned,FixedPoint.Unsigned,uint256).maxCollateralPerToken (packages/core/contract
/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#218) is too similar to PerpetualLiquidatablePoolParty.createLiquidation(address,FixedPoint.Unsigned,Fixe
Point.Unsigned,FixedPoint.Unsigned,uint256).minCollateralPerToken (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
resetMember(uint256,address) should be declared external:
        - MultiRole.resetMember(uint256,address) (packages/core/contracts/common/implementation/MultiRole.sol#130-133)
getMember(uint256) should be declared external:
        - MultiRole.getMember(uint256) (packages/core/contracts/common/implementation/MultiRole.sol#141-143)
addMember(uint256,address) should be declared external:
        - MultiRole.addMember(uint256,address) (packages/core/contracts/common/implementation/MultiRole.sol#152-155)
removeMember(uint256,address) should be declared external:
        - MultiRole.removeMember(uint256,address) (packages/core/contracts/common/implementation/MultiRole.sol#164-167)
renounceMembership(uint256) should be declared external:
        - MultiRole.renounceMembership(uint256) (packages/core/contracts/common/implementation/MultiRole.sol#175-178)
getCurrentTime() should be declared external:
        - Timer.getCurrentTime() (packages/core/contracts/common/implementation/Timer.sol#28-30)
pfc() should be declared external:
        - FeePayerPoolParty.pfc() (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#124-126)
collateralCurrency() should be declared external:
        - FeePayerPoolParty.collateralCurrency() (packages/core/contracts/financial-templates/common/FeePayerPoolParty.sol#132-134)
isMinter(address) should be declared external:
        - MintableBurnableSyntheticToken.isMinter(address) (packages/core/contracts/financial-templates/common/MintableBurnableSyntheticToken.sol#93-95)
isBurner(address) should be declared external:
        - MintableBurnableSyntheticToken.isBurner(address) (packages/core/contracts/financial-templates/common/MintableBurnableSyntheticToken.sol#102-104)
isAdmin(address) should be declared external:
        - MintableBurnableSyntheticToken.isAdmin(address) (packages/core/contracts/financial-templates/common/MintableBurnableSyntheticToken.sol#111-113)
withdrawLiquidation(uint256,address) should be declared external:
        - PerpetualLiquidatablePoolParty.withdrawLiquidation(uint256,address) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#30
-324)
deploy(PerpetualLiquidatablePoolParty.ConstructorParams) should be declared external:
        - PerpetualPoolPartyLib.deploy(PerpetualLiquidatablePoolParty.ConstructorParams) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolPartyLib.sol#
9-22)
deposit(FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.deposit(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#241-
44)
withdraw(FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.withdraw(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#253-
265)
requestWithdrawal(FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.requestWithdrawal(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPart
.sol#272-282)
create(FixedPoint.Unsigned,FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.create(FixedPoint.Unsigned,FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManage
PoolParty.sol#322-332)
redeem(FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.redeem(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#343-3
1)
repay(FixedPoint.Unsigned) should be declared external:
        - PerpetualPositionManagerPoolParty.repay(FixedPoint.Unsigned) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#370-38
)
createPerpetual(PerpetualPoolPartyCreator.Params) should be declared external:
        - PerpetualPoolPartyCreator.createPerpetual(PerpetualPoolPartyCreator.Params) (packages/core/contracts/financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.so
#71-106)
getRoleAdmin(bytes32) should be declared external:
        - AccessControl.getRoleAdmin(bytes32) (packages/core/contracts/openzeppelin/contracts/access/AccessControl.sol#105-107)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (packages/core/contracts/openzeppelin/contracts/access/AccessControl.sol#134-138)
owner() should be declared external:
        - Ownable.owner() (packages/core/contracts/openzeppelin/contracts/access/Ownable.sol#33-35)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (packages/core/contracts/openzeppelin/contracts/access/Ownable.sol#52-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (packages/core/contracts/openzeppelin/contracts/access/Ownable.sol#61-65)
name() should be declared external:
        - ERC20.name() (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (packages/core/contracts/openzeppelin/contracts/token/ERC20/ERC20.sol#89-91)
```



MythX

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analysers to locate any vulnerabilities. Security Detections are only in scope, and the analysis was pointed towards issues with these PerpetualLiquidatablePoolParty.sol, PerpetualLiquidatablePoolPartyLib.sol, PerpetualPoolParty.sol, PerpetualPoolPartyLib.sol, PerpetualPositionManagerPoolParty.sol, PerpetualPositionManagerPoolPartyLib.sol, PerpetutalPoolPartyCreator.sol contracts .

1. PerpetutalPoolPartyCreator.sol

Report for common/implementation/MultiRole.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
130	(SWC-000) Unknown	Medium	Function could be marked as external.
141	(SWC-000) Unknown	Medium	Function could be marked as external.
152	(SWC-000) Unknown	Medium	Function could be marked as external.
164	(SWC-000) Unknown	Medium	Function could be marked as external.
175	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for common/implementation/Timer.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
28	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for financial-templates/common/FeePayerPoolParty.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
124	(SWC-000) Unknown	Medium	Function could be marked as external.
132	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for financial-templates/common/MintableBurnableSyntheticToken.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
29	(SWC-000) Unknown	Medium	Function could be marked as external.
38	(SWC-000) Unknown	Medium	Function could be marked as external.
47	(SWC-000) Unknown	Medium	Function could be marked as external.
56	(SWC-000) Unknown	Medium	Function could be marked as external.
63	(SWC-000) Unknown	Medium	Function could be marked as external.
70	(SWC-000) Unknown	Medium	Function could be marked as external.
77	(SWC-000) Unknown	Medium	Function could be marked as external.
84	(SWC-000) Unknown	Medium	Function could be marked as external.
93	(SWC-000) Unknown	Medium	Function could be marked as external.
102	(SWC-000) Unknown	Medium	Function could be marked as external.
111	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
305	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
241	(SWC-000) Unknown	Medium	Function could be marked as external.
253	(SWC-000) Unknown	Medium	Function could be marked as external.
272	(SWC-000) Unknown	Medium	Function could be marked as external.
322	(SWC-000) Unknown	Medium	Function could be marked as external.
343	(SWC-000) Unknown	Medium	Function could be marked as external.
370	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for financial-templates/perpetual-poolParty/PerpetutalPoolPartyCreator.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
71	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for oracle/interfaces/OracleInterface.sol
https://dashboard.mythx.io/#/console/analyses/4f5e8fa6-c8e9-4ee7-9069-11dcee2daaed

Line	SWC Title	Severity	Short Description
15	(SWC-000) Unknown	Medium	Function could be marked as external.
24	(SWC-000) Unknown	Medium	Function could be marked as external.
33	(SWC-000) Unknown	Medium	Function could be marked as external.

2. PerpetualPoolPartyLib.sol

Report for financial-templates/perpetual-poolParty/PerpetualPoolPartyLib.sol
https://dashboard.mythx.io/#/console/analyses/6b82f66d-00b3-4d1b-ac89-4265fc750c90

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

3. PerpetualPoolParty.sol

Report for financial-templates/perpetual-poolParty/PerpetualPoolParty.sol
https://dashboard.mythx.io/#/console/analyses/df44c95c-6dcb-44ef-9122-c9c4c5bf4e0f

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

4. PerpetualLiquidatablePoolParty.sol

Report for financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol
https://dashboard.mythx.io/#/console/analyses/b727831c-7015-4c7f-85c5-182da84f17ed

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
305	(SWC-000) Unknown	Medium	Function could be marked as external.

5. PerpetualLiquidatablePoolPartyLib.sol

Report for financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol
https://dashboard.mythx.io/#/console/analyses/df1451dc-d941-4f10-a4cc-e51837dee9a4

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

6. PerpetualPositionManagerPoolPartyLib.sol

Report for financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol
https://dashboard.mythx.io/#/console/analyses/33cb42e3-15a0-410f-91ea-f3d2e9f8153a

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

7. PerpetualPositionManagerPoolParty.sol

Report for financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol
https://dashboard.mythx.io/#/console/analyses/33cb42e3-15a0-410f-91ea-f3d2e9f8153a

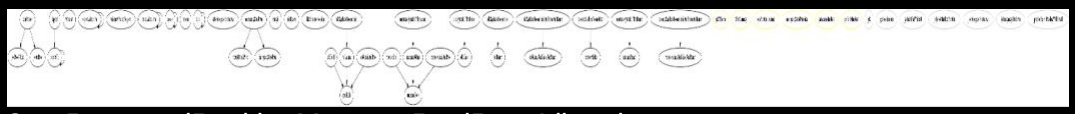
Line	SWC Title	Severity	Short Description
241	(SWC-000) Unknown	Medium	Function could be marked as external.
253	(SWC-000) Unknown	Medium	Function could be marked as external.
272	(SWC-000) Unknown	Medium	Function could be marked as external.
322	(SWC-000) Unknown	Medium	Function could be marked as external.
343	(SWC-000) Unknown	Medium	Function could be marked as external.
370	(SWC-000) Unknown	Medium	Function could be marked as external.

3.5 SOLGRAPH - INFORMATIONAL

Description:
Solgraph tools runs on all possible scoped solidity contracts to get display the control flow dependencies and identify security flaw.

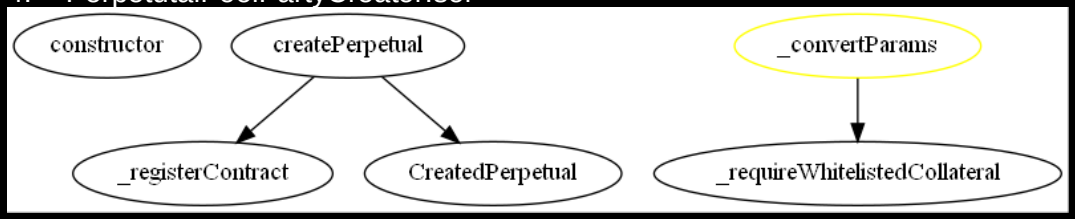
1. PerpetualLiquidatablePoolPartyLib.sol

2. PerpetualPositionManagerPoolParty.sol



3. PerpetualPositionManagerPoolPartyLib.sol

4. PerpetutalPoolPartyCreator.sol



THANK YOU FOR CHOOSING

// HALBORN