



JARVIS

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 09, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY	3
CONTACTS	3
1 EXECUTIVE SUMMARY	4
1.1 INTRODUCTION	5
1.2 TEST APPROACH & METHODOLOGY	5
1.3 SCOPE	6
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	7
3 FINDINGS & TECH DETAILS	7
3.1 FLOATING PRAGMA - LOW	9
Description	9
Code Location	9
Recommendations	9
3.2 MISSING ADDRESS CHECK - LOW	9
Description	9
Code Location	10
Recommendation	10
3.3 USE OF BLOCK.TIMESTAMP - INFORMATIONAL	11
Description	11
Code Location	11
Recommendation	11
3.4 IGNORE RETURN VALUES - INFORMATIONAL	12
Description	12
Code Location	12
Recommendation	13

3.5 STATIC ANALYSIS REPORT	14
Description	14
Results	14
3.6 AUTOMATED SECURITY SCAN	17
Description	17
Results	18

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/09/2021	Nishit Majithia

CONTACTS

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

EXECUTIVE SUMMARY

1.1 INTRODUCTION

JARVIS engaged Halborn to conduct a security assessment on their all smart contracts including Oracle and Synthereum (the protocol behind Jarvis Exchange). The security assessment was scoped to the smart contracts under these folders `contracts/base/`, `contracts/derivative/`, `contracts/oracle/`, `contracts/synthereum-pool/v3/` and `contracts/versioning/`. 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 contract code is well documented, follows a high-quality software development standard, contains 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.

1.2 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](#))

1.3 SCOPE

IN-SCOPE:

- The security assessment was scoped to the smart contracts under these folders `contracts/base/`, `contracts/derivative/`, `contracts/oracle/`, `contracts/synthetix-pool/v3/` and `contracts/versioning/`.

Specific commit of contract: commit

`27f7c257038b349352a0bdfcafba4a1c35fd569d`

OUT-OF-SCOPE:

Other smart contracts in the repository, customized uma-fork, external libraries and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
FLOATING PRAGMA	Low	-
MISSING ADDRESS CHECK	Low	-
USE OF BLOCK.TIMESTAMP	Informational	-
IGNORE RETURN VALUES	Informational	-
STATIC ANALYSIS	-	-
AUTOMATED SECURITY SCAN	-	-



FINDINGS & TECH DETAILS



3.1 FLOATING PRAGMA - LOW

Description:

All Smart Contracts use the floating pragma ^0.6.12. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an updated compiler version that might introduce bugs or discovered vulnerabilities in the newest versions that affect the contract system negatively.

Code Location:

Every contract under the scope is using floating pragma.

Recommendations:

Consider lock the pragma version known bugs for the compiler version. When possible, do not use floating pragma in the final live deployment. Pragma can also be locked fixing the compiler version in the configuration file when you deploy contracts with truffle or hardhat frameworks.

3.2 MISSING ADDRESS CHECK - LOW

Description:

Address validation at some places in contracts `contracts/derivative/v1/DerivativeFactory.sol`, `contracts/derivative/v1/SyntheticTokenFactory.sol` and `contracts/synthetix-pool/v3/PoolChainPriceFeedFactory.sol`. Lack of zero address validation has been found at many instances when assigning user supplied address values to state variables directly.

Code Location:

contracts/derivative/v1/DerivativeFactory.sol:

Line #30

```
21  constructor(
22    address _synthereumFinder,
23    address _umaFinder,
24    address _tokenFactoryAddress,
25    address _timerAddress
26  )
27  public
28  PerpetualPoolPartyCreator(_umaFinder, _tokenFactoryAddress, _timerAddress)
29 {
30   synthereumFinder = _synthereumFinder;
31   deploymentSignature = this.createPerpetual.selector;
32 }
```

contracts/derivative/v1/SyntheticTokenFactory.sol:

Line #22

```
20
21
22  constructor(address _synthereumFinder, uint8 _derivativeVersion) public {
23    synthereumFinder = _synthereumFinder;
24    derivativeVersion = _derivativeVersion;
25
26  function createToken(
```

contracts/synthereum-pool/v3/PoolChainPriceFeedFactory.sol:

Line #38

```
33 /**
34  * @notice Set synthereum finder
35  * @param _synthereumFinder Synthereum finder contract
36  */
37  constructor(address _synthereumFinder) public {
38    synthereumFinder = _synthereumFinder;
39    deploymentSignature = this.createPool.selector;
40  }
41
42  //-----
43  // Public functions
44  //
```

Recommendation:

Add proper address validation when every state variable assignment done from user supplied input.

3.3 USE OF BLOCK.TIMESTAMP - INFORMATIONAL

Description:

Block timestamps have historically been used for a variety of applications, such as entropy for random numbers, locking funds for periods of time and various state-changing conditional statements that are time-dependent. Miner's have the ability to adjust timestamps slightly which can prove to be quite dangerous if block timestamps are used incorrectly in smart contracts.

`block.timestamp` or it's alias `now` can be manipulated by miners if they have some incentive to do so

Code Location:

`contracts/synthetix-pool/v3/PoolChainPriceFeedLib.sol`:

```
Line #1134
1128     function checkParams(
1129         ISynthetixPoolOnChainPriceFeedStorage.Storage storage self,
1130         IDerivative derivative,
1131         uint256 feePercentage,
1132         uint256 expiration
1133     ) internal view checkDerivative(self, derivative) {
1134         require(now <= expiration, 'Transaction expired');
1135         require(
1136             self.fee.feePercentage.rawValue <= feePercentage,
1137             'User fee percentage less than actual one'
1138         );
1139     }
```

Recommendation:

Avoid getting relied on use of `block.timestamp` in require methods in order to prevent possible miners manipulation.

3.4 IGNORE RETURN VALUES - INFORMATIONAL

Description:

The return value of an external call is not stored in a local or state variable. In contracts `contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`, `contracts/versioning/PoolRegister.sol` and `contracts/versioning/FactoryVersioning.sol`, there are few instances where external methods are being called and return value(bool) are being ignored.

Code Location:

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`:

```
Line #178
170     , EXCEN 1
171     self.version = _version;
172     self.finder = _finder;
173     self.startingCollateralization = _startingCollateralization;
174     self.isContractAllowed = _isContractAllowed;
175     self.collateralToken = getDerivativeCollateral(_derivative);
176     self.syntheticToken = _derivative.tokenCurrency();
177     self.priceIdentifier = _derivative.positionManagerData().priceIdentifier;
178     self.derivatives.add(address(_derivative));
179     emit AddDerivative(address(this), address(_derivative));
180   }
181 }
```

Line #596

```
593   }
594
595   // Redeem the synthetic tokens for collateral
596   derivative.settleEmergencyShutdown();
597
598   // Amount of collateral that will be redeemed and sent to the user
599   FixedPoint.Unsigned memory totalToRedeem;
600
601   // If the user is the LP, send redeemed token collateral plus excess collateral
602   if (isLiquidityProvider) {
```

Line #719

```

717         finder.getImplementationAddress(SynthereumInterfaces.PoolRegistry)
718     );
719     poolRegister.isPoolDeployed(
720         pool.syntheticTokenSymbol(),
721         collateralToken,
722         pool.version(),
723         address(pool)
724     );
725     if (
726         derivativeRole == ISynthereumPoolOnChainPriceFeed.DerivativeRoles.POOL
727     ) {

```

contracts/versioning/PoolRegister.sol:

Line #40-41

```

36     ISynthereumFinder(synthereumFinder).getImplementationAddress(
37         SynthereumInterfaces.Deployer
38     );
39     require(msg.sender == deployer, 'Sender must be Synthereum deployer');
40     symbolToPools[syntheticTokenSymbol][collateralToken][poolVersion].add(pool);
41     collaterals.add(address(collateralToken));
42 }
43
44 function isPoolDeployed(
45     string calldata poolSymbol,
46     IERC20 collateral,

```

contracts/versioning/FactoryVersioning.sol

Line #66

Line #77

```

64
65     function removePoolFactory(uint8 version) external override onlyMaintainer {
66     require(
67         _poolsFactory.remove(version),
68         'Version of the pool factory does not exist'
69     );
70     emit RemovePoolFactory(version);
71 }
72
73     function setDerivativeFactory(uint8 version, address derivativeFactory)
74     external
75     override
76     onlyMaintainer
77     {
78         _derivativeFactory.set(version, derivativeFactory);
79         emit AddDerivativeFactory(version, derivativeFactory);
80     }

```

Recommendation:

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

3.5 STATIC ANALYSIS REPORT

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. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

```

INFO:Detectors:
PerpetualLiquidatablePoolPartyLib (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#11-567) contract sets array length with a user-controlled value:
    - liquidations.punPerpetualLiquidatablePoolParty.liquidationData(params.sponsor,msg.sender,PerpetualLiquidatablePoolParty_Status.PreDispute,params.actualTime,returnValues.tokensLiquidated,returnValues.lockedInLiquidity,returnValues.coreContractAddress,FixedPoint.fromScaledUint(0),FixedPoint.fromScaledUint(0),returnValues.finalFeeBps) (@jarvis-network/uma-core/contracts/oracle/implementation/Registry.sol#155-174)
Registry (@jarvis-network/uma-core/contracts/oracle/implementation/Registry.sol#109) contract sets array length with a user-controlled value:
    - partyMap[party].contracts.push(contractAddress) (@jarvis-network/uma-core/contracts/oracle/implementation/Registry.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
INFO:Detectors:
PerpetualPositionManagerPoolParty_positions (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#68) is a public mapping with nested variables
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-mappings-with-nested-variables
THREE.js:
INFO:Detectors:
Reentrancy in PerpetualPositionManagerPoolParty.create(FixedPoint.Unsigned,fixedPoint.Unsigned) (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#169-210)
External calls:
    - fees() (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#266)
        - returnData = address(token).functionCall(data,SafeERC20_low-level call failed) (@openzeppelin/contracts/token/ERC20/SafeERC20.sol#79-80)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/utils/Address.sol#65)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/financial-templates/common/feePayerPoolParty.sol#69)
        - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#66-68)
        - store.payOracleFeeSrC20(address(feePayerData.collateralCurrency),regularFee.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#70-73)
        - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#77-80)
External calls (including eth):
    - fees() (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#266)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/utils/Address.sol#65)
State variables written after the call(s):
    - nonReentrant = false (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#28)
        - _notEntered = false (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#28)
        - _notEntered = true (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#32)
Reentrancy in PerpetualLiquidatablePoolParty.createLiquidation(address,FixedPoint.Unsigned,FixedPoint.Unsigned,uint256) (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#125-127)
External calls:
    - fees() (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#166)
        - returnData = address(token).functionCall(data,SafeERC20_low-level call failed) (@openzeppelin/contracts/token/ERC20/SafeERC20.sol#79-80)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/financial-templates/common/feePayerPoolParty.sol#69)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/utils/Address.sol#65)
        - feePayerData.collateralCurrency.safeIncreaseAllowance(address(store),regularFee.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#66-68)
        - store.payOracleFeeSrC20(address(feePayerData.collateralCurrency),regularFee.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#70-73)
        - feePayerData.collateralCurrency.safeTransfer(msg.sender,latePenalty.rawValue) (@jarvis-network/uma-core/contracts/Financial-Templates/common/feePayerPoolPartyLib.sol#77-80)
External calls (including eth):
    - fees() (@jarvis-network/uma-core/contracts/Financial-Templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#166)
        - (success,returnData) = target.call(value: value)(data) (@openzeppelin/contracts/utils/Address.sol#65)
State variables written after the call(s):
    - nonReentrant = false (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#28)
        - _notEntered = false (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#28)
        - _notEntered = true (@jarvis-network/uma-core/contracts/common/implementation/Lockable.sol#32)

```

- These issues highlighted by slither is in Jarvis's UMA-fork or in openzeppelin library, which is out of the scope for this Audit. But this issue has been already covered in Halborn's security audit of Jarvis's UMA-Fork smart contracts.

```

INFO:Detectors:
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData)
PerpetualLiquidatablePoolParty.LiquidatableData.FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#469-472)
    -lockedCollateral = liquidationCollateralParams.tokensLiquidated.div(positionOfLiquidate.tokensOutstanding) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#469-472)
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData)
PerpetualLiquidatablePoolParty.LiquidatableData.FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#452-458) performs a multiplication on the result of a division
    -ratio = liquidationCollateralParams.tokensLiquidated.div(positionOfLiquidate.tokensOutstanding) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#452-458)
PerpetualLiquidatablePoolPartyLib.liquidateCollateral(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData)
PerpetualLiquidatablePoolParty.LiquidatableData.FeePayerPoolParty.FeePayerData,PerpetualLiquidatablePoolPartyLib.CreateLiquidationCollateral) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#452-458) performs a multiplication on the result of a division
    -ratio = liquidationCollateralParams.tokensLiquidated.div(positionOfLiquidate.tokensOutstanding) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#452-458)
    -withdrawalAmountToRemove = positionToLiquidate.withdrawRequestAmount.mul(ratio) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#458-481)
PerpetualPositionManagerPoolPartyLib.redememt(PerpetualPositionManagerPoolParty.PositionData,PerpetualPositionManagerPoolParty.GlobalPositionData,PerpetualPositionManagerPoolParty.PositionManagerData,FixedPoint.Unsigned,FeePayerPoolParty.FeePayerData,address) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#272-336) performs a multiplication on t e result of a division:
    -fractionRedeemed = numTokens.div(positionData.tokensOutstanding) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#287-288)
    -collateralReceived = fractionRedeemed.mul(positionData.rawCollateral.getFeeAdjustedCollateral(FeePayerData.cumulativeFeeMultiplier)) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolPartyLib.sol#288-294)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PerpetualPositionManagerPoolParty._notEmergencyShutdown() (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#529-534) uses a dangerous strict equality:
    - require(bool,string) positionManagerData.emergencyShutdownTimestamp == 0,Contract emergency shutdown (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#530-533)
PerpetualPositionManagerPoolParty.emergencyShutdown() (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#326-346) uses a dangerous strict equality:
    - require(bool,string) hasRole(POOL_ROLE,msg.sender) || msg.sender == _getFinancialContractsAdminAddress(),Caller must be a pool or the UMA governor (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#327-330)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
PerpetualLiquidatablePoolPartyLib.withdrawLiquidation(PerpetualLiquidatablePoolParty.LiquidationData,PerpetualLiquidatablePoolParty.LiquidatableData,PerpetualPositionManagerPoolParty.PositionManagerData,FeePayerPoolParty.FeePayerData,uint256,address).settleParams (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolPartyLib.sol#284) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

```

- Again above Divide Before Multiply and Dangerous Strict Equalities issues are in the Jarvis's UMA-Fork library and out of scope for this audit.

```

INFO:Detectors:
SyntherumPoolRegistry.registerPool(string,IERC20,uint8,address) (contracts/versioning/PoolRegister.sol#29-42) ignores return value by symbolToPools[syntheticTokenSymbol][collateralToken][poolVersion].add(pool)
SyntherumPoolRegistry.registerPool(string,IERC20,uint8,address) (contracts/versioning/PoolRegister.sol#29-42) ignores return value by collaterals.add(address(collateralToken)) (contracts/versioning/PoolRegister.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
SyntherumFactoryVersioning.setPoolFactory(uint8,address) (contracts/versioning/FactoryVersioning.sol#55-62) ignores return value by _poolsFactory.set(version,poolFactory) (contracts/versioning/FactoryVersioning.sol#60-69)
SyntherumFactoryVersioning.setDerivativeFactory(uint8,address) (contracts/versioning/FactoryVersioning.sol#72-79) ignores return value by _derivativeFactory.set(version,derivativeFactory) (contracts/versioning/FactoryVersioning.sol#77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
SyntherumPoolOnChainPriceFeedLib.initialize(ISyntherumPoolOnChainPriceFeedStorage.Storage,uint8,ISyntherumFinder,IDerivative,FixedPoint.Unsigned,bool) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#163-180) ignores return value by self.derivatives.add(address(derivative)) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#178)
SyntherumPoolOnChainPriceFeedLib.setEmergencyShutdown(ISyntherumPoolOnChainPriceFeedStorage.IDerivative,bytes32) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#560-629) ignores return value by self.emergencyShutdown(ISyntherumPoolOnChainPriceFeedStorage.IDerivative,bytes32) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#560-629)
SyntherumPoolOnChainPriceFeedLib.addRoleInDerivative(ISyntherumPoolOnChainPriceFeedStorage.IDerivative,ISyntherumPoolOnChainPriceFeed.DerivativeRoles,address) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#692-736) ignores return value by poolRegister.isPoolDeployed(pool.syntheticTokenSymbol),collateralToken,pool.version(),address(pool)) (contracts/syntherum-pool/v3/PoolOnChainPriceFeedLib.sol#719-774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

```

- These issues have already mentioned in the analysis report.

```

INFO:Detectors:
PerpetualPositionManagerPoolParty.constructor(PerpetualPositionManagerPoolParty.PositionManagerParams,PerpetualPositionManagerPoolParty.Roles)._roles (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#19)
    - AccessControl.Roles(_openzeppelin/contracts/access/AccessControl.sol#17) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
PerpetualPoolPartyCreator.constructor(address,address,address),_tokenfactoryAddress (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolPartyCreator.sol#43) lacks a zero-check on :
    - tokenfactoryAddress = _tokenfactoryAddress (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPoolPartyCreator.sol#51)
        - syntherumFinder = _syntherumFinder (contracts/derivative/v1/DerivativeFactory.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PerpetualLiquidatablePoolParty._disputable(uint256,address) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#296-304) uses timestamp for comparisons
    - Dangerous comparisons:
        - require(bool,string)(getCurrentTime() < _getLiquidationExpiry(liquidation)) && (liquidation.state == Status.PreDispute),Liquidation not disputable (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#290-303)
    PerpetualLiquidatablePoolParty._withdrawable(uint256,address) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#306-317) uses timestamp for comparisons
        - Dangerous comparisons:
            - require(bool,string)((state > Status.PreDispute) || ((getLiquidationExpiry(liquidation) < getCurrentTime()) && (state == Status.PreDispute)),Liquidation not withdrawable) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualLiquidatablePoolParty.sol#306-317)
    PerpetualLiquidatablePoolParty._emergencyShutdown() (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#326-346) uses timestamp for comparisons
        - require(bool,string)(hasRole(POOL_ROLE,msg.sender) || msg.sender == _getFinancialContractsAdminAddress(),Caller must be a pool or the UMA governor (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#332-336)
    PerpetualPositionManagerPoolParty._onlyCollateralizedPosition(address) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#18519-527) uses timestamp for comparisons
        - Dangerous comparisons:
            - require(bool,string)(positions(sponsor).rawCollateral.getFeeAdjustedCollateral(FeePayerData.cumulativeFeeMultiplier).isGreaterThan(0),Position has no collateral) (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#526-528)
    PerpetualPositionManagerPoolParty._notEmergencyShutdown() (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#529-534) uses timestamp for comparisons
        - Dangerous comparisons:
            - require(bool,string)(positionManagerData.emergencyShutdownTimestamp == 0,Contract emergency shutdown (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#530-533)
    PerpetualPositionManagerPoolParty._isEmergencyShutdown() (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#536-541) uses timestamp for comparisons
        - Dangerous comparisons:
            - require(bool,string)(positionManagerData.emergencyShutdownTimestamp != 0,Contract not emergency shutdown (@jarvis-network/uma-core/contracts/financial-templates/perpetual-poolParty/PerpetualPositionManagerPoolParty.sol#537-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

```

- Above issues are related to Jarvis's UMA-Fork, which were out of the scope for this audit.

```

INFO:Detectors:
SynthereumSyntheticTokenFactory.constructor(address,uint8)_synthereumFinder (contracts/derivative/v1/SyntheticTokenFactory.sol#21) lacks a zero-check on :
- synthereumFinder = _synthereumFinder (contracts/derivative/v1/SyntheticTokenFactory.sol#22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address._isContract(address) (@openzeppelin/contracts/utils/Address.sol#5-12) uses assembly
- Address._isContract (contracts/assembly/Address.sol#10) (@openzeppelin/contracts/utils/Address.sol#89-106) uses assembly
Address._verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#89-106) uses assembly
- INLINE ASM (@openzeppelin/contracts/utils/Address.sol#98-101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
- Version used: ['>=0.6.0<0.8.0', '>0.6.2<0.8.0', '^0.6.0', '^0.6.12']
  - 0.6.0 (@j�rvis-network/uma-core/contracts/common/implementations/MintableBurnableERC20.sol#2)
    - 0.6.0 (@j�rvis-network/uma-core/contracts/common/interfaces/MintableBurnableERC20.sol#3)
  - 0.6.0 (@j�rvis-network/uma-core/contracts/financial-templates/common/MintableBurnableSyntheticToken.sol#2)
  - 0.6.0 (@j�rvis-network/uma-core/contracts/financial-templates/common/MintableBurnableTokenFactory.sol#2)
  - 0.6.0 (@j�rvis-network/uma-core/contracts/AccessControl.sol#2)
  - 0.6.0 (@openzeppelin/contracts/math/SafeMath.sol#2)
  - 0.6.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#2)
  - 0.6.0 (@openzeppelin/contracts/contract/IERC20.sol#2)
  - 0.6.0 (@openzeppelin/contracts/utils/Address.sol#2)
  - 0.6.0 (@openzeppelin/contracts/utils/EnumerableSet.sol#2)
  - 0.6.12 (contracts/derivative/v1/SyntheticTokenFactory.sol#2)
  - ABIEncoderV2 (contracts/derivative/v1/SyntheticTokenFactory.sol#3)
  - 0.6.12 (contracts/versioning/Constants.sol#2)
  - 0.6.12 (contracts/versioning/interfaces/IFactoryVersioning.sol#2)
  - 0.6.12 (contracts/versioning/interfaces/IFinder.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

```

- This missing zero-address validation issues have already mentioned in the analysis report.

```

INFO:Detectors:
SynthereumChainlinkPriceFeed.constructor(ISynthereumFinder,SynthereumChainlinkPriceFeed.Roles)_roles (contracts/oracle/chainlink/ChainlinkPriceFeed.sol#61) shadows:
- AccessControl._roles (@openzeppelin/contracts/access/AccessControl.sol#17) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SynthereumFinder.constructor(SynthereumFinder.Roles)_roles (contracts/versioning/Finder.sol#3) shadows:
- AccessControl._roles (@openzeppelin/contracts/access/AccessControl.sol#17) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

```

- Local variable shadowing is not impacting here since `AccessssControl._roles` variable is not used by contract. So it is false positive.

```

INFO:Detectors:
Pragmas version 0.6.12 (contracts/versioning/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc 0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
SynthereumPoolOnChainPriceFeedLib.checkParams(ISynthereumPoolOnChainPriceFeedStorage,IDerivative,uint256,uint256) (contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol#1128-1139) uses timestamp for comparisons:
- Dangerous comparisons:
  - require(block.timestamp)(<now && expirationTransaction_expired) (contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol#134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SynthereumPoolOnChainPriceFeedFactory.constructor(address)_synthereumFinder (contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol#37) lacks a zero-check on :
- synthereumFinder = _synthereumFinder (contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

```

- These issues have already mentioned in the analysis report.

3.6 AUTOMATED SECURITY SCAN

Description:

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 analyzers to locate any vulnerabilities. In addition, Security Detections are only in scope.

Results:

`contracts/synthereum-pool/v3/PoolOnChainPriceFeed.sol`

Report for `contracts/synthereum-pool/v3/PoolOnChainPriceFeed.sol`
<https://dashboard.mythx.io/#/console/analyses/345e4951-7571-4156-a131-c594314599c5>

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

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Report for `contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`
<https://dashboard.mythx.io/#/console/analyses/345e4951-7571-4156-a131-c594314599c5>

Line	SWC Title	Severity	Short Description
141	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.
1128	(SWC-000) Unknown	Medium	Incorrect function "checkParams" state mutability

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol`

Report for `contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol`
<https://dashboard.mythx.io/#/console/analyses/25351a22-87e8-4cb2-ba01-e354a2e020ad>

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

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedCreator.sol`

Report for `contracts/synthereum-pool/v3/PoolOnChainPriceFeedCreator.sol`
<https://dashboard.mythx.io/#/console/analyses/52dcbcf8-cdeb-4e70-9b75-9f9c035b750b>

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

`contracts/versioning/Constants.sol`

Report for `contracts/versioning/Constants.sol`
<https://dashboard.mythx.io/#/console/analyses/27af861e-484f-4c28-add1-6bd69d4a8f1e>

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

contracts/versioning/Deployer.sol

Report for contracts/versioning/Deployer.sol
<https://dashboard.mythx.io/#/console/analyses/d0394098-2ddc-4e1a-bc87-570b1ee612b2>

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

contracts/oracle/chainlink/ChainlinkPriceFeed.sol

Report for contracts/oracle/chainlink/ChainlinkPriceFeed.sol
<https://dashboard.mythx.io/#/console/analyses/eefbf8c69-c1fa-4eaa-82d2-ffcf4f59638c>

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
82	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.

contracts/derivative/v1/SyntheticTokenFactory.sol

Report for contracts/derivative/v1/SyntheticTokenFactory.sol
<https://dashboard.mythx.io/#/console/analyses/d21d2c1f-c0ff-4837-92a7-c62453fe482a>

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

contracts/derivative/v1/DerivativeFactory.sol

Report for contracts/derivative/v1/DerivativeFactory.sol
<https://dashboard.mythx.io/#/console/analyses/178e7679-99ce-49a1-97e7-8107b07151d6>

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

THANK YOU FOR CHOOSING
HALBORN