PROJECT

Rysk Gamma Protocol

CLIENT

Rysk Finance

DATE

July 2022

REVIEWERS

Andrei Simion

@andreiashu

Table of Contents

Details

- **Client** Rysk Finance
- **Date** July 2022
- **Lead reviewer** Andrei Simion (@andreiashu)
- **Repository**: Rysk Gamma Protocol
- **Commit hash** `5eab7d899cac06daadbed7c46462d58b778a2775`
- **Technologies**
  - Solidity
  - Typescript

Issues Summary

SEVERITY	OPEN	CLOSED
Informational	1	0
Minor	1	0
Medium	1	0
Major	0	0

Executive summary

This report represents the results of the engagement with **Rysk Finance** to review **Rysk Gamma Protocol** - a fork of Opyn's Gamma Protocol.

The review was conducted over the course of **1 week** from **July 18 to July 25, 2022**. A total of **6 person-days** were spent reviewing the code.

Scope

The initial review focused on the Rysk Gamma Protocol repository, identified by the commit hash `5eab7d899cac06daadbed7c46462d58b778a2775`. The Rysk team made changes to Opyn's Gamma Protocol to allow collateral in USDC for *CALL* options and ETH for *PUT* options. The previous auction system for liquidating vaults was removed, and a more direct liquidation mechanism was implemented in its place.

The changes that were made are included in a pull request against a fork of Gamma Protocol within the Rysk Finance Github Organisation: https://github.com/rysk-finance/GammaProtocol/pull/1

I focused on manually reviewing the codebase, searching for security issues such as, but not limited to, re-entrancy problems, transaction ordering, block timestamp dependency, exception handling, call stack depth limitation, integer overflow/underflow, self-destructible contracts, unsecured balance, use of origin, costly gas patterns, architectural problems, code readability.

Includes:

- code/contracts/core/Controller.sol
- code/contracts/core/Whitelist.sol
- code/contracts/core/MarginCalculator.sol
- code/contracts/libs/MarginVault.sol
- code/contracts/core/AddressBook.sol
- code/contracts/pricers/ChainlinkPricer.sol

- code/contracts/core/Otoken.sol
- code/contracts/core/Oracle.sol
- code/contracts/core/MarginPool.sol
- code/contracts/core/OtokenFactory.sol
- code/contracts/core/OtokenSpawner.sol
- code/contracts/libs/Actions.sol

Recommendations

I identified a few possible general improvements that are not security issues during the review, which will bring value to the developers and the community reviewing and using the product.

Increase the number of tests

A good rule of thumb is to have 100% test coverage. This does not guarantee the lack of security problems, but it means that the desired functionality behaves as intended. The negative tests also bring a lot of value because not allowing some actions to happen is also part of the desired behavior.

Issues

`ChainLinkPricer.getPrice` might return outdated or invalid pricing data

Status Open Severity Medium

Description

Vaults liquidation calculation happens based on the asset's current price. This change was made as part of the Rysk pull request to the Opyn Gamma protocol:

code/contracts/core/MarginCalculator.sol#L438-L457

```
    function isLiquidatable(MarginVault.Vault memory _vault, uint256 _vaultType)
        external
        view
        returns (
            bool,
            uint256,
            uint256
        )
    {
```

```
        // liquidation is only supported for naked margin vault
        require(_vaultType == 1, "MarginCalculator: invalid vault type to liquidate");

        VaultDetails memory vaultDetails = _getVaultDetails(_vault, _vaultType);

        // can not liquidate vault that have no short position
        if (!vaultDetails.hasShort) return (false, 0, 0);

        require(now < vaultDetails.shortExpiryTimestamp, "MarginCalculator: can not liquidate expired posi

        uint256 price = oracle.getPrice(vaultDetails.shortUnderlyingAsset);
```

However, the code handling Chainlink's price feed does not check whether the data might be stale:

code/contracts/pricers/ChainlinkPricer.sol#L83-L87

```
    function getPrice() external view override returns (uint256) {
        (, int256 answer, , , ) = aggregator.latestRoundData();
        require(answer > 0, "ChainLinkPricer: price is lower than 0");
        // chainlink's answer is already 1e8
        return _scaleToBase(uint256(answer));
```

A complete code to validate data returned by Chainlink's `latestRoundData` would include checking against a few other attributes:

```
        (uint80 roundId, int256 answer, , uint256 timestamp, uint80 answeredInRound ) = aggregator.late
         require(answer > 0, "ChainLinkPricer: price is lower than 0");
         require(timestamp != 0, "ROUND_NOT_COMPLETE");
         require(block.timestamp <= timestamp + stalePriceDelay, "STALE_PRICE");
         require(answeredInRound >= roundId, "STALE_PRICE");
```

The `stalePriceDelay` parameter should be configured on a per-pair basis since different data feeds have various guarantees for how long ago the last answer was committed to the blockchain.

This is also what Chainlink's documentation in their v0.7 version of the code, which some pair data feeds still run on:

```
/**
 * @notice get data about the latest round. Consumers are encouraged to check
 * that they're receiving fresh data by inspecting the updatedAt and
 * answeredInRound return values.
 * Note that different underlying implementations of AggregatorV3Interface
 * have slightly different semantics for some of the return values. Consumers
 * should determine what implementations they expect to receive
 * data from and validate that they can properly handle return data from all
```

```
    * of them.
```

Options for handling stale or invalid price feed data

Currently, there is a missing piece of logic with regards to the behavior of the liquidation system if price data returned by Chainlink is stale. At the moment, the code will either:

1. continue working on a stale price as long as the value returned is greater than 0:

code/contracts/pricers/ChainlinkPricer.sol#L85

```
        require(answer > 0, "ChainLinkPricer: price is lower than 0");
```

2. If the suggested fixes are implemented, the code will revert when Chainlink returns stale data. This behavior is still suboptimal since it puts the existing vaults at risk of not being liquidated in time.

Below we explore some options that can be implemented in future versions to handle this scenario.

1. Redundant price feed sources

One option for a more resilient price feed data source is to have redundancy and support a fallback price feed. This is what Liquity project does:

> Liquity functions that require the most current ETH:USD price data fetch the price dynamically, as needed, via the core PriceFeed.sol contract using the Chainlink ETH:USD reference contract as its primary and Tellor's ETH:USD price feed as its secondary (fallback) data source.

Liquity, once deployed on a network, is immutable (there are no config values to be updated nor proxy contracts to allow for the logic to be swapped). Therefore, in their case, this added complexity is valuable:

> We believe the benefit of the fallback logic is worth the complexity, given that our system is entirely immutable - if we had no fallback logic and Chainlink were to be hacked or permanently fail, Liquity would become permanently unusable anyway.

In the case of Rysk's Gamma Protocol, the ability for the contract to automatically use a fallback pricing feed platform needs to be weighed against the added complexity and additional developer investment required to deliver such functionality.

2. Allow for a manual data source as a fallback

Another option that can be considered is to allow for a *bot* like role account that can push pricing data to the Rysk Gamma storage in case there is an issue with the Chainlink data feeds. The contract code would check this "local" data feed **only if** the

Chainlink oracle request reverts or returns invalid data (Solidity's try/catch feature can help here).

This is the least preferred option because of the obvious complexities of maintaining a high uptime and additional security risks of having an off-chain, centralized data source. However, the security risk is somewhat mitigated by the fact that this data source will be relevant only in cases where the Chainlink data feed is invalid.

Recommendation

Given that Rysk will use Chainlink pairs that are already well established (like ETH/USD), the risk of the Chainlink data feed being invalid (for long enough periods) is very small.

The best way forward is to add a redundancy layer to the pricing data feed. That being said, I do not see value in stopping a release of the Rysk platform based on this requirement.

References

OZ: Smart Contract Security Guidelines #3: The Dangers of Price Oracles

Solidity 0.6.x features: try/catch statement

`Oracle.setStablePrice` should not allow for `0` price value

Status Open Severity Minor

Description

`Oracle.setStablePrice` allows the owner to set the price value for a stablecoin:

code/contracts/core/Oracle.sol#L157-L162

```
    function setStablePrice(address _asset, uint256 _price) external onlyOwner {
        require(assetPricer[_asset] == address(0), "Oracle: could not set stable price for an asset with p

        stablePrice[_asset] = _price;

        emit StablePriceUpdated(_asset, _price);
```

However, a value of `0` would not make sense in this case since the system would stall because `getPrice` function would revert:

code/contracts/core/Oracle.sol#L216-L220

```
    function getPrice(address _asset) external view returns (uint256) {
```

```
        uint256 price = stablePrice[_asset];


        if (price == 0) {
            require(assetPricer[_asset] != address(0), "Oracle: Pricer for this asset not set");
```

Recommendation

Validate that `_price` is greater than `0` in `setStablePrice` function.

`Controller._isLiquidatable` unused local variable `latestUpdateTimestamp`

Status Open Severity Informational

Description

`latestUpdateTimestamp` can be removed since it's not used in the code following its declaration:

code/contracts/core/Controller.sol#L1119-L1122

```
        (MarginVault.Vault memory vault, uint256 typeVault, uint256 latestUpdateTimestamp) = getVaultWithD
            _owner,
            _vaultId
        );
```

Artifacts

Sūrya's Description Report

Sūrya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Files Description Table

File Name	SHA-1 Hash
code/contracts/core/Controller.sol	f452149ba24655c8d44721cdf797a6d51ef2
code/contracts/core/Whitelist.sol	615ec322652a2746910623ca8415c1eb471
code/contracts/core/MarginCalculator.sol	843b3fcbe45906343f57d3189f48b1d7ee6a
code/contracts/libs/MarginVault.sol	ac2db58d7430be80d38cbfa058dccc0812c
code/contracts/core/AddressBook.sol	b99fbdf9f548e4ca5f04e155090011a7716e3
code/contracts/pricers/ChainlinkPricer.sol	453da1e6a7cc64e15ee297de3157db733c7
code/contracts/core/Otoken.sol	8e10c8069e4dc8dfeea9d58758bc72f2912
code/contracts/core/Oracle.sol	8a842959055b739aa8284fc7664909cb075
code/contracts/core/MarginPool.sol	9f98e5150051badf0a36dd5bfd3cabdf3ecf8
code/contracts/core/OtokenFactory.sol	bb085179c4acdd3caaa327be44664188743
code/contracts/core/OtokenSpawner.sol	bab114a27fa6e70e63934c575675f417da9b
code/contracts/libs/Actions.sol	4f5310b55174d5457790fe8347be0c1166d

Contracts Description Table

Contract	Type	Bases
└	**Function Name**	**Visibility**
Controller	Implementation	Initializable, OwnableUpgradeSafe, ReentrancyGuardUpgradeSafe
└	_isNotPartiallyPaused	Internal 🔒
└	_isNotFullyPaused	Internal 🔒
└	_isAuthorized	Internal 🔒
└	initialize	External ❗
└	donate	External ❗
└	setSystemPartiallyPaused	External ❗
└	setSystemFullyPaused	External ❗
└	setFullPauser	External ❗
└	setPartialPauser	External ❗
└	setCallRestriction	External ❗
└	setOperator	External ❗
└	refreshConfiguration	External ❗
└	setNakedCap	External ❗
└	operate	External ❗
└	sync	External ❗
└	clearVaultLiquidationDetails	External ❗
└	isOperator	External ❗
└	getConfiguration	External ❗
└	getProceed	External ❗
└	getVaultLiquidationDetails	External ❗
└	isLiquidatable	External ❗
└	getPayout	Public ❗
└	isSettlementAllowed	External ❗

Contract	Type	Bases
└	canSettleAssets	External !
└	getAccountVaultCounter	External !
└	hasExpired	External !
└	getVault	External !
└	getVaultWithDetails	Public !
└	getNakedCap	External !
└	getNakedPoolBalance	External !
└	_runActions	Internal 🔒
└	_verifyFinalState	Internal 🔒
└	_openVault	Internal 🔒
└	_depositLong	Internal 🔒
└	_withdrawLong	Internal 🔒
└	_depositCollateral	Internal 🔒
└	_withdrawCollateral	Internal 🔒
└	_mintOtoken	Internal 🔒
└	_burnOtoken	Internal 🔒
└	_redeem	Internal 🔒
└	_settleVault	Internal 🔒
└	_liquidate	Internal 🔒
└	_call	Internal 🔒
└	_checkVaultId	Internal 🔒
└	_isNotEmpty	Internal 🔒

Contract	Type	Bases
└	_isCalleeWhitelisted	Internal 🔒
└	_isLiquidatable	Internal 🔒
└	_getOtokenDetails	Internal 🔒
└	_canSettleAssets	Internal 🔒
└	_refreshConfigInternal	Internal 🔒
Whitelist	Implementation	Ownable
└		Public ❗
└	isWhitelistedProduct	External ❗
└	isWhitelistedCollateral	External ❗
└	isCoveredWhitelistedCollateral	External ❗
└	isNakedWhitelistedCollateral	External ❗
└	isWhitelistedOtoken	External ❗
└	isWhitelistedCallee	External ❗
└	whitelistProduct	External ❗
└	blacklistProduct	External ❗
└	whitelistCollateral	External ❗
└	whitelistCoveredCollateral	External ❗
└	whitelistNakedCollateral	External ❗
└	blacklistCollateral	External ❗
└	whitelistOtoken	External ❗
└	blacklistOtoken	External ❗
└	whitelistCallee	External ❗
└	blacklistCallee	External ❗
MarginCalculator	Implementation	Ownable
└		Public ❗
└	setCollateralDust	External ❗
└	setLiquidationMultiplier	External ❗
└	setUpperBoundValues	External ❗

Contract	Type	Bases
└	updateUpperBoundValue	External !
└	setSpotShock	External !
└	setOracleDeviation	External !
└	getCollateralDust	External !
└	getTimesToExpiry	External !
└	getMaxPrice	External !
└	getSpotShock	External !
└	getOracleDeviation	External !
└	getNakedMarginRequired	External !
└	getExpiredPayoutRate	External !
└	isLiquidatable	External !
└	getMarginRequired	External !
└	getExcessCollateral	Public !
└	_getExpiredCashValue	Internal 🔒
└	_getMarginRequired	Internal 🔒
└	_getNakedMarginRequired	Internal 🔒
└	_findUpperBoundValue	Internal 🔒
└	_getPutSpreadMarginRequired	Internal 🔒
└	_getCallSpreadMarginRequired	Internal 🔒
└	_convertAmountOnLivePrice	Internal 🔒
└	_convertAmountOnExpiryPrice	Internal 🔒
└	_getDebtPrice	Internal 🔒
└	_getVaultDetails	Internal 🔒
└	_getExpiredSpreadCashValue	Internal 🔒
└	_isNotEmpty	Internal 🔒
└	_checkIsValidVault	Internal 🔒
└	_isMarginableLong	Internal 🔒
└	_isMarginableCollateral	Internal 🔒
└	_getProductHash	Internal 🔒

Contract	Type	Bases
└	_getCashValue	Internal 🔒
└	_getOtokenDetails	Internal 🔒
└	makeShortScaledDetails	Internal 🔒
└	getOptionType	Internal 🔒
MarginVault	Library	
└	addShort	External ❗
└	removeShort	External ❗
└	addLong	External ❗
└	removeLong	External ❗
└	addCollateral	External ❗
└	removeCollateral	External ❗
AddressBook	Implementation	Ownable
└	getOtokenImpl	External ❗
└	getOtokenFactory	External ❗
└	getWhitelist	External ❗
└	getController	External ❗
└	getMarginPool	External ❗
└	getMarginCalculator	External ❗
└	getLiquidationManager	External ❗
└	getOracle	External ❗
└	setOtokenImpl	External ❗
└	setOtokenFactory	External ❗
└	setWhitelist	External ❗
└	setController	External ❗
└	setMarginPool	External ❗
└	setMarginCalculator	External ❗
└	setLiquidationManager	External ❗
└	setOracle	External ❗

Contract	Type	Bases
└	getAddress	Public ❗
└	setAddress	Public ❗
└	updateImpl	Public ❗
ChainLinkPricer	Implementation	OpynPricerInterface
└		Public ❗
└	setExpiryPriceInOracle	External ❗
└	getPrice	External ❗
└	getHistoricalPrice	External ❗
└	_scaleToBase	Internal 🔒
Otoken	Implementation	ERC20PermitUpgradeable
└	init	External ❗
└	getOtokenDetails	External ❗
└	mintOtoken	External ❗
└	burnOtoken	External ❗
└	_getNameAndSymbol	Internal 🔒
└	_getDisplayedStrikePrice	Internal 🔒
└	_uintTo2Chars	Internal 🔒
└	_getOptionType	Internal 🔒
└	_slice	Internal 🔒
└	_getMonth	Internal 🔒
Oracle	Implementation	Ownable
└	migrateOracle	External ❗
└	endMigration	External ❗
└	setAssetPricer	External ❗
└	setLockingPeriod	External ❗
└	setDisputePeriod	External ❗
└	setDisputer	External ❗
└	setStablePrice	External ❗

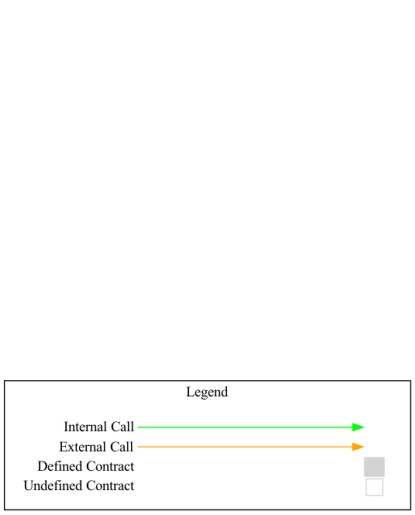
Contract	Type	Bases
└	disputeExpiryPrice	External ❗
└	setExpiryPrice	External ❗
└	getPrice	External ❗
└	getExpiryPrice	External ❗
└	getPricer	External ❗
└	getDisputer	External ❗
└	getPricerLockingPeriod	External ❗
└	getPricerDisputePeriod	External ❗
└	getChainlinkRoundData	External ❗
└	isLockingPeriodOver	Public ❗
└	isDisputePeriodOver	Public ❗
MarginPool	Implementation	Ownable
└		Public ❗
└	transferToPool	Public ❗
└	transferToUser	Public ❗
└	getStoredBalance	External ❗
└	batchTransferToPool	External ❗
└	batchTransferToUser	External ❗
└	farm	External ❗
└	setFarmer	External ❗
OtokenFactory	Implementation	OtokenSpawner
└		Public ❗
└	createOtoken	External ❗
└	getOtokensLength	External ❗
└	getOtoken	External ❗
└	getTargetOtokenAddress	External ❗
└	_getOptionId	Internal 🔒
OtokenSpawner	Implementation	

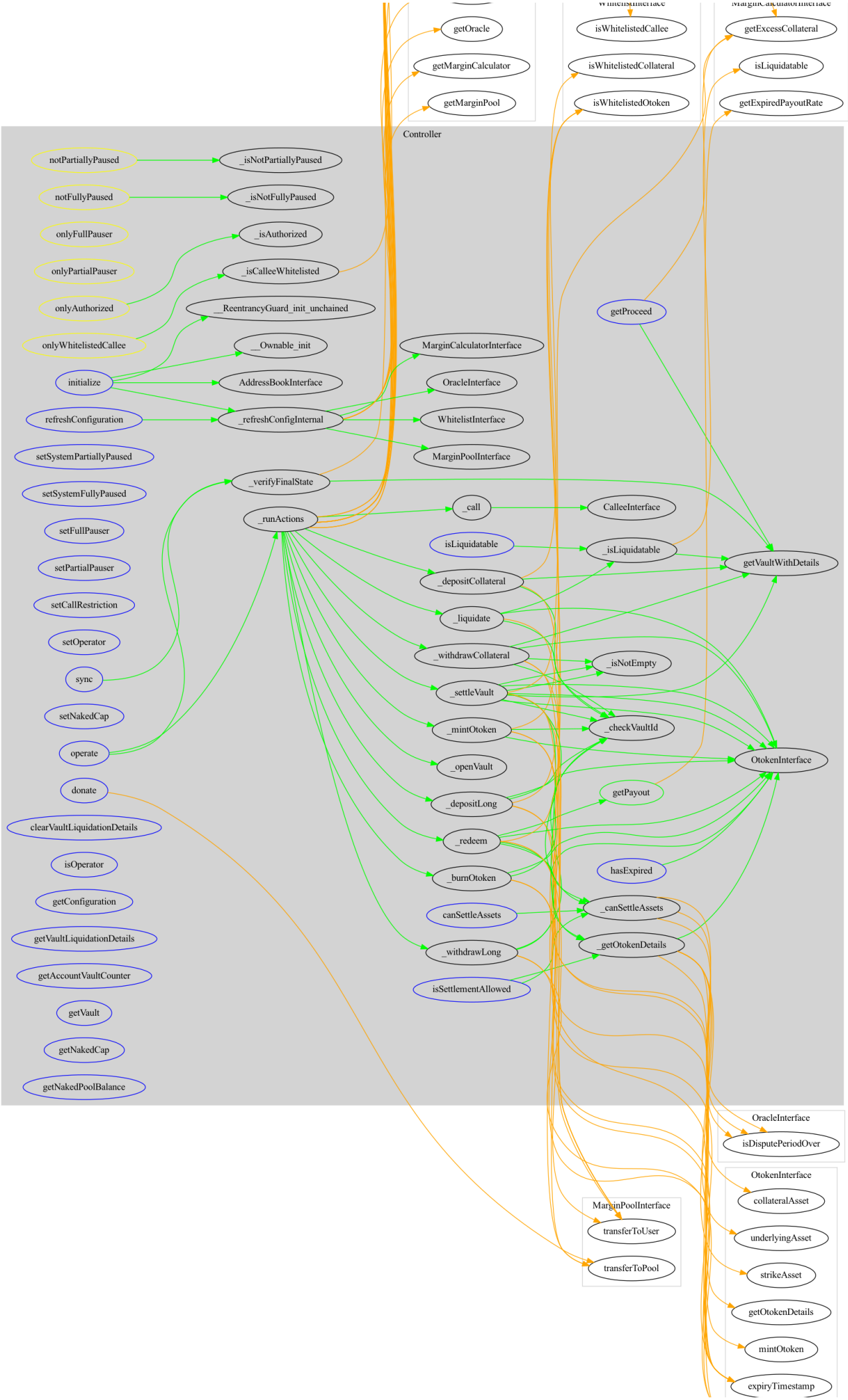
Contract	Type	Bases
└	_spawn	Internal 🔒
└	_computeAddress	Internal 🔒
Actions	Library	
└	_parseOpenVaultArgs	Internal 🔒
└	_parseMintArgs	Internal 🔒
└	_parseBurnArgs	Internal 🔒
└	_parseDepositArgs	Internal 🔒
└	_parseWithdrawArgs	Internal 🔒
└	_parseRedeemArgs	Internal 🔒
└	_parseSettleVaultArgs	Internal 🔒
└	_parseLiquidateArgs	Internal 🔒
└	_parseCallArgs	Internal 🔒

Legend

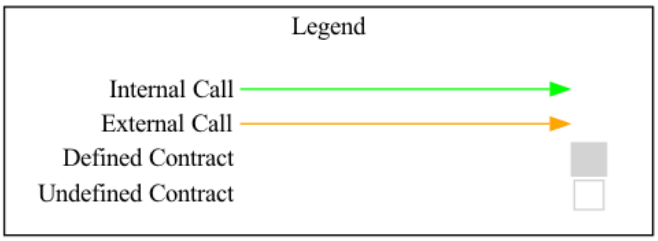
Symbol	Meaning
	Function can modify state
	Function is payable

Graphs

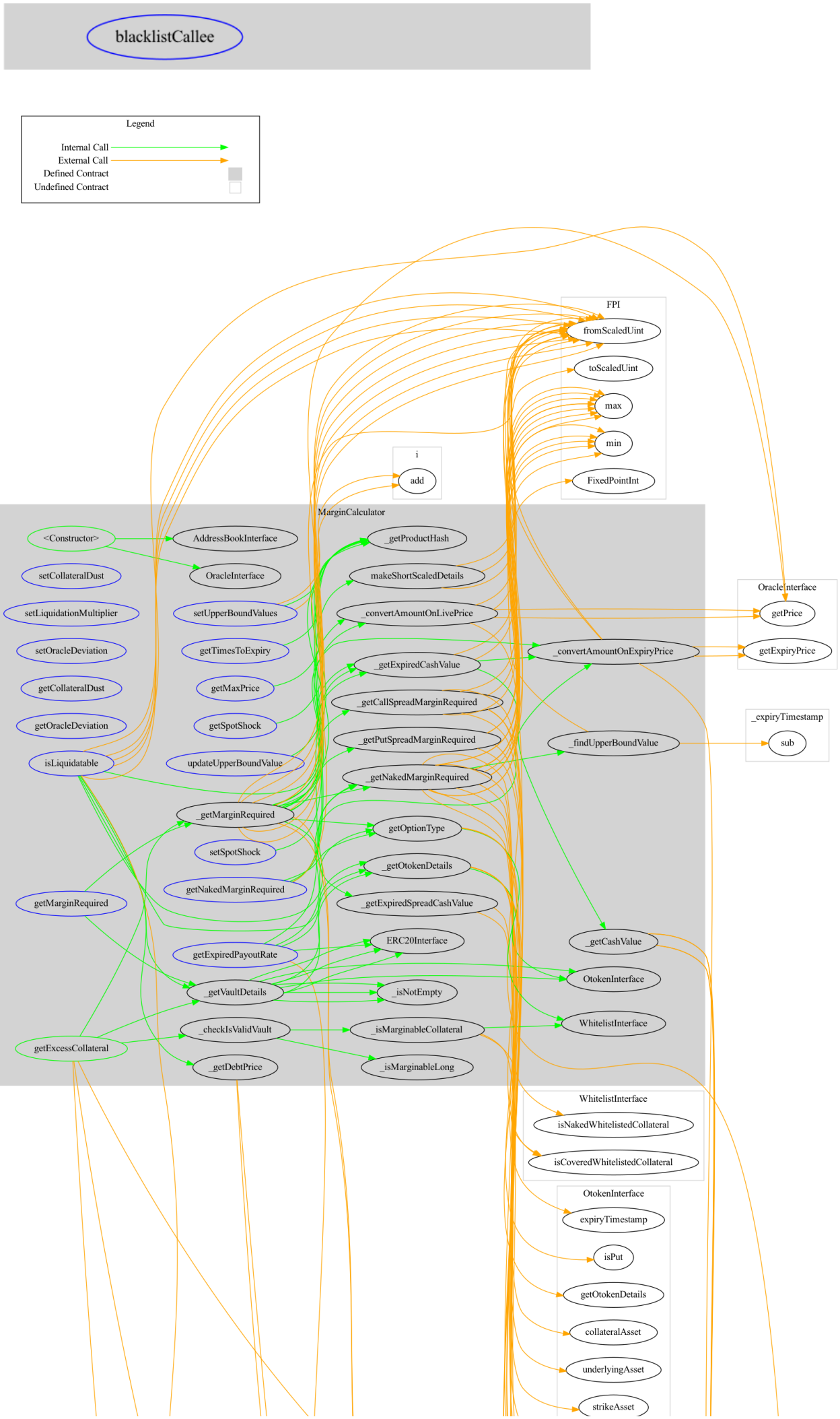


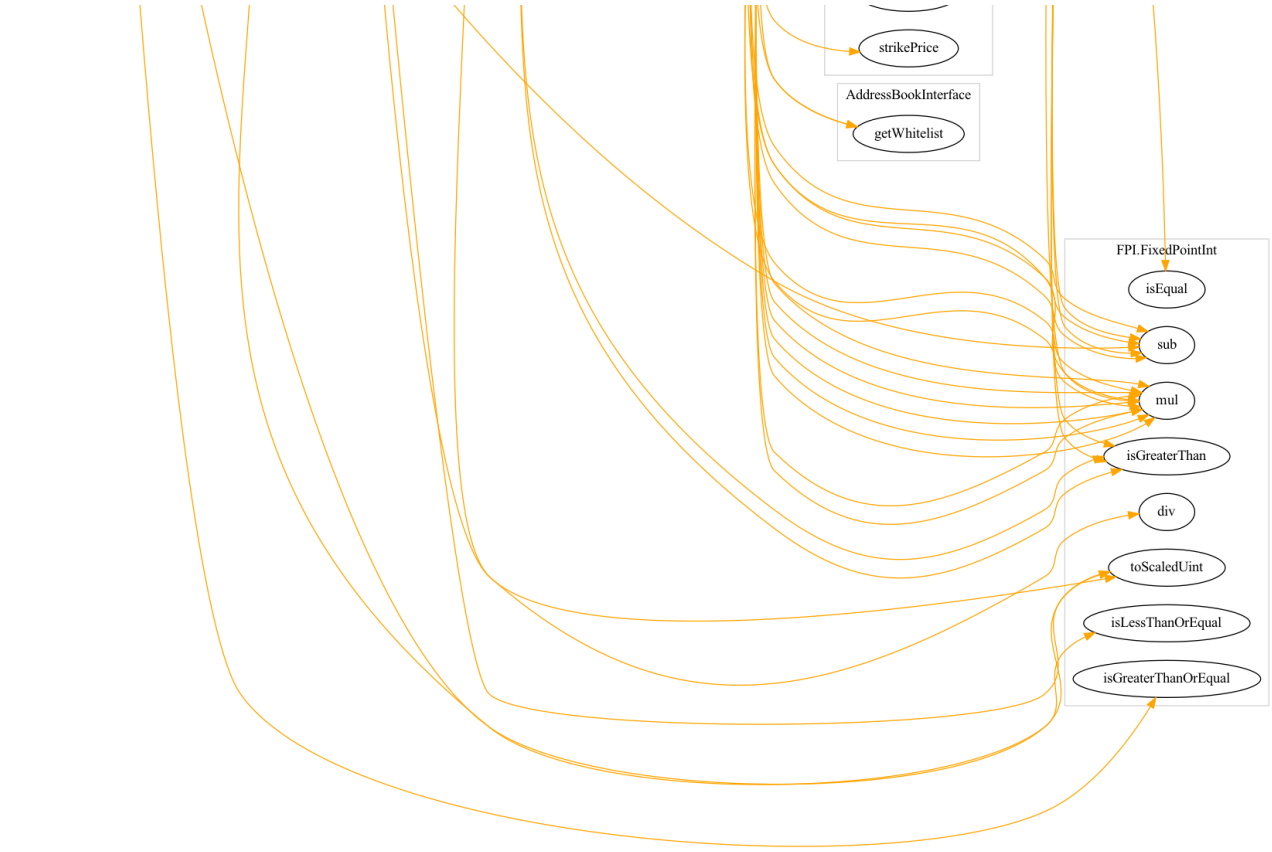

getOracle
getMarginCalculator
getMarginPool
isWhitelistedCallee
isWhitelistedCollateral
isWhitelistedOtoken
getExcessCollateral
isLiquidatable
getExpiredPayoutRate
Controller
notPartiallyPaused
_isNotPartiallyPaused
notFullyPaused
_isNotFullyPaused
onlyFullPauser
_isAuthorized
onlyPartialPauser
_isCalleeWhitelisted
onlyAuthorized
__ReentrancyGuard_init_unchained
onlyWhitelistedCallee
__Ownable_init
initialize
AddressBookInterface
refreshConfiguration
_refreshConfigInternal
setSystemPartiallyPaused
setSystemFullyPaused
setFullPauser
setPartialPauser
setCallRestriction
setOperator
sync
setNakedCap
operate
donate
clearVaultLiquidationDetails
isOperator
getConfiguration
getVaultLiquidationDetails
getAccountVaultCounter
getVault
getNakedCap
getNakedPoolBalance
MarginCalculatorInterface
OracleInterface
WhitelistInterface
MarginPoolInterface
_verifyFinalState
_runActions
_call
CalleeInterface
isLiquidatable
_isLiquidatable
_depositCollateral
_liquidate
_withdrawCollateral
_settleVault
_mintOtoken
_openVault
_depositLong
_redeem
_burnOtoken
canSettleAssets
_withdrawLong
isSettlementAllowed
getProceed
_isNotEmpty
_checkVaultId
OtokenInterface
getPayout
hasExpired
_canSettleAssets
_getOtokenDetails
getVaultWithDetails
OracleInterface
isDisputePeriodOver
OtokenInterface
collateralAsset
underlyingAsset
strikeAsset
getOtokenDetails
mintOtoken
expiryTimestamp
MarginPoolInterface
transferToUser
transferToPool

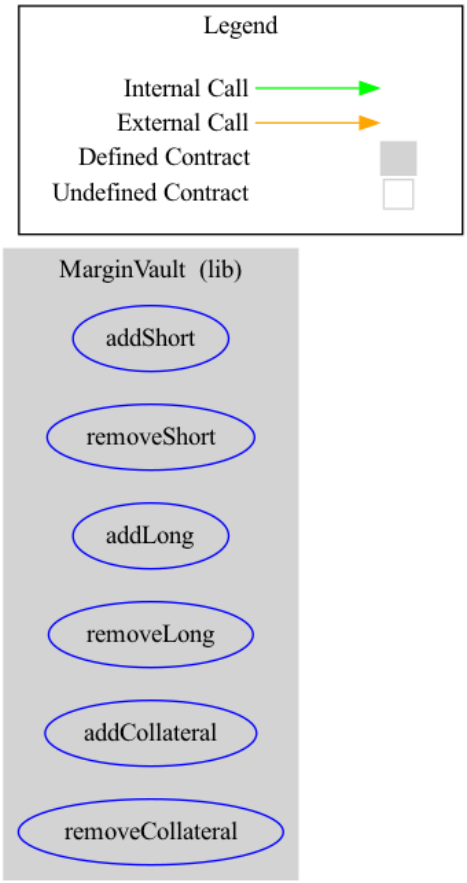

burnOtoken


Legend
Internal Call
External Call
Defined Contract
Undefined Contract


Whitelist
<Constructor>
onlyFactory
AddressBookInterface
isWhitelistedProduct
isWhitelistedCollateral
isCoveredWhitelistedCollateral
isNakedWhitelistedCollateral
isWhitelistedOtoken
isWhitelistedCallee
whitelistProduct
blacklistProduct
whitelistCollateral
whitelistCoveredCollateral
whitelistNakedCollateral
blacklistCollateral
whitelistOtoken
blacklistOtoken
whitelistCallee


blacklistCallee
Legend
Internal Call
External Call
Defined Contract
Undefined Contract
FPI
fromScaledUint
toScaledUint
max
min
FixedPointInt
i
add
MarginCalculator
<Constructor>
AddressBookInterface
OracleInterface
setCollateralDust
setLiquidationMultiplier
setOracleDeviation
getCollateralDust
getOracleDeviation
isLiquidatable
setUpperBoundValues
getTimesToExpiry
getMaxPrice
getSpotShock
updateUpperBoundValue
_getMarginRequired
setSpotShock
getNakedMarginRequired
getMarginRequired
getExpiredPayoutRate
_getVaultDetails
_checkIsValidVault
getExcessCollateral
_getDebtPrice
_getProductHash
makeShortScaledDetails
_convertAmountOnLivePrice
_getExpiredCashValue
_getCallSpreadMarginRequired
_getPutSpreadMarginRequired
_getNakedMarginRequired
getOptionType
_getOtokenDetails
_getExpiredSpreadCashValue
ERC20Interface
_isNotEmpty
_isMarginableCollateral
_isMarginableLong
OracleInterface
getPrice
getExpiryPrice
_convertAmountOnExpiryPrice
_findUpperBoundValue
_expiryTimestamp
sub
_getCashValue
OtokenInterface
WhitelistInterface
WhitelistInterface
isNakedWhitelistedCollateral
isCoveredWhitelistedCollateral
OtokenInterface
expiryTimestamp
isPut
getOtokenDetails
collateralAsset
underlyingAsset
strikeAsset





Describe

```
$ npx surya describe code/contracts/core/Controller.sol code/contracts/core/Whitelist.sol code/contracts/c
```

```
+  Controller (Initializable, OwnableUpgradeSafe, ReentrancyGuardUpgradeSafe)
   - [Int] _isNotPartiallyPaused
   - [Int] _isNotFullyPaused
   - [Int] _isAuthorized
   - [Ext] initialize #
      - modifiers: initializer
   - [Ext] donate #
   - [Ext] setSystemPartiallyPaused #
      - modifiers: onlyPartialPauser
   - [Ext] setSystemFullyPaused #
      - modifiers: onlyFullPauser
   - [Ext] setFullPauser #
      - modifiers: onlyOwner
   - [Ext] setPartialPauser #
      - modifiers: onlyOwner
   - [Ext] setCallRestriction #
      - modifiers: onlyOwner
   - [Ext] setOperator #
   - [Ext] refreshConfiguration #
      - modifiers: onlyOwner
   - [Ext] setNakedCap #
      - modifiers: onlyOwner
   - [Ext] operate #
      - modifiers: nonReentrant,notFullyPaused
   - [Ext] sync #
      - modifiers: nonReentrant,notFullyPaused
   - [Ext] clearVaultLiquidationDetails #
   - [Ext] isOperator
   - [Ext] getConfiguration
   - [Ext] getProceed
   - [Ext] getVaultLiquidationDetails
   - [Ext] isLiquidatable
   - [Pub] getPayout
   - [Ext] isSettlementAllowed
   - [Ext] canSettleAssets
   - [Ext] getAccountVaultCounter
   - [Ext] hasExpired
   - [Ext] getVault
   - [Pub] getVaultWithDetails
   - [Ext] getNakedCap
   - [Ext] getNakedPoolBalance
   - [Int] _runActions #
   - [Int] _verifyFinalState
   - [Int] _openVault #
      - modifiers: notPartiallyPaused,onlyAuthorized
   - [Int] _depositLong #
      - modifiers: notPartiallyPaused,onlyAuthorized
   - [Int] _withdrawLong #
      - modifiers: notPartiallyPaused,onlyAuthorized
   - [Int] _depositCollateral #
      - modifiers: notPartiallyPaused,onlyAuthorized
```

```
    - [Int] _withdrawCollateral #
       - modifiers: notPartiallyPaused,onlyAuthorized
    - [Int] _mintOtoken #
       - modifiers: notPartiallyPaused,onlyAuthorized
    - [Int] _burnOtoken #
       - modifiers: notPartiallyPaused,onlyAuthorized
    - [Int] _redeem #
    - [Int] _settleVault #
       - modifiers: onlyAuthorized
    - [Int] _liquidate #
       - modifiers: notPartiallyPaused
    - [Int] _call #
       - modifiers: notPartiallyPaused,onlyWhitelistedCallee
    - [Int] _checkVaultId
    - [Int] _isNotEmpty
    - [Int] _isCalleeWhitelisted
    - [Int] _isLiquidatable
    - [Int] _getOtokenDetails
    - [Int] _canSettleAssets
    - [Int] _refreshConfigInternal #

+   Whitelist (Ownable)
    - [Pub] <Constructor> #
    - [Ext] isWhitelistedProduct
    - [Ext] isWhitelistedCollateral
    - [Ext] isCoveredWhitelistedCollateral
    - [Ext] isNakedWhitelistedCollateral
    - [Ext] isWhitelistedOtoken
    - [Ext] isWhitelistedCallee
    - [Ext] whitelistProduct #
       - modifiers: onlyOwner
    - [Ext] blacklistProduct #
       - modifiers: onlyOwner
    - [Ext] whitelistCollateral #
       - modifiers: onlyOwner
    - [Ext] whitelistCoveredCollateral #
       - modifiers: onlyOwner
    - [Ext] whitelistNakedCollateral #
       - modifiers: onlyOwner
    - [Ext] blacklistCollateral #
       - modifiers: onlyOwner
    - [Ext] whitelistOtoken #
       - modifiers: onlyFactory
    - [Ext] blacklistOtoken #
       - modifiers: onlyOwner
    - [Ext] whitelistCallee #
       - modifiers: onlyOwner
    - [Ext] blacklistCallee #
       - modifiers: onlyOwner

+   MarginCalculator (Ownable)
```

```
    - [Pub] <Constructor> #
    - [Ext] setCollateralDust #
       - modifiers: onlyOwner
    - [Ext] setLiquidationMultiplier #
       - modifiers: onlyOwner
    - [Ext] setUpperBoundValues #
       - modifiers: onlyOwner
    - [Ext] updateUpperBoundValue #
       - modifiers: onlyOwner
    - [Ext] setSpotShock #
       - modifiers: onlyOwner
    - [Ext] setOracleDeviation #
       - modifiers: onlyOwner
    - [Ext] getCollateralDust
    - [Ext] getTimesToExpiry
    - [Ext] getMaxPrice
    - [Ext] getSpotShock
    - [Ext] getOracleDeviation
    - [Ext] getNakedMarginRequired
    - [Ext] getExpiredPayoutRate
    - [Ext] isLiquidatable
    - [Ext] getMarginRequired
    - [Pub] getExcessCollateral
    - [Int] _getExpiredCashValue
    - [Int] _getMarginRequired
    - [Int] _getNakedMarginRequired
    - [Int] _findUpperBoundValue
    - [Int] _getPutSpreadMarginRequired
    - [Int] _getCallSpreadMarginRequired
    - [Int] _convertAmountOnLivePrice
    - [Int] _convertAmountOnExpiryPrice
    - [Int] _getDebtPrice
    - [Int] _getVaultDetails
    - [Int] _getExpiredSpreadCashValue
    - [Int] _isNotEmpty
    - [Int] _checkIsValidVault
    - [Int] _isMarginableLong
    - [Int] _isMarginableCollateral
    - [Int] _getProductHash
    - [Int] _getCashValue
    - [Int] _getOtokenDetails
    - [Int] makeShortScaledDetails
    - [Int] getOptionType

+ [Lib] MarginVault
    - [Ext] addShort #
    - [Ext] removeShort #
    - [Ext] addLong #
    - [Ext] removeLong #
    - [Ext] addCollateral #
    - [Ext] removeCollateral #
```

```
+  AddressBook (Ownable)
   - [Ext] getOtokenImpl
   - [Ext] getOtokenFactory
   - [Ext] getWhitelist
   - [Ext] getController
   - [Ext] getMarginPool
   - [Ext] getMarginCalculator
   - [Ext] getLiquidationManager
   - [Ext] getOracle
   - [Ext] setOtokenImpl #
      - modifiers: onlyOwner
   - [Ext] setOtokenFactory #
      - modifiers: onlyOwner
   - [Ext] setWhitelist #
      - modifiers: onlyOwner
   - [Ext] setController #
      - modifiers: onlyOwner
   - [Ext] setMarginPool #
      - modifiers: onlyOwner
   - [Ext] setMarginCalculator #
      - modifiers: onlyOwner
   - [Ext] setLiquidationManager #
      - modifiers: onlyOwner
   - [Ext] setOracle #
      - modifiers: onlyOwner
   - [Pub] getAddress
   - [Pub] setAddress #
      - modifiers: onlyOwner
   - [Pub] updateImpl #
      - modifiers: onlyOwner

+  ChainLinkPricer (OpynPricerInterface)
   - [Pub] <Constructor> #
   - [Ext] setExpiryPriceInOracle #
      - modifiers: onlyBot
   - [Ext] getPrice
   - [Ext] getHistoricalPrice
   - [Int] _scaleToBase

+  Otoken (ERC20PermitUpgradeable)
   - [Ext] init #
      - modifiers: initializer
   - [Ext] getOtokenDetails
   - [Ext] mintOtoken #
   - [Ext] burnOtoken #
   - [Int] _getNameAndSymbol
   - [Int] _getDisplayedStrikePrice
   - [Int] _uintTo2Chars
   - [Int] _getOptionType
   - [Int] _slice
```

```
    - [Int] _getMonth

+  Oracle (Ownable)
    - [Ext] migrateOracle #
       - modifiers: onlyOwner
    - [Ext] endMigration #
       - modifiers: onlyOwner
    - [Ext] setAssetPricer #
       - modifiers: onlyOwner
    - [Ext] setLockingPeriod #
       - modifiers: onlyOwner
    - [Ext] setDisputePeriod #
       - modifiers: onlyOwner
    - [Ext] setDisputer #
       - modifiers: onlyOwner
    - [Ext] setStablePrice #
       - modifiers: onlyOwner
    - [Ext] disputeExpiryPrice #
    - [Ext] setExpiryPrice #
    - [Ext] getPrice
    - [Ext] getExpiryPrice
    - [Ext] getPricer
    - [Ext] getDisputer
    - [Ext] getPricerLockingPeriod
    - [Ext] getPricerDisputePeriod
    - [Ext] getChainlinkRoundData
    - [Pub] isLockingPeriodOver
    - [Pub] isDisputePeriodOver

+  MarginPool (Ownable)
    - [Pub] <Constructor> #
    - [Pub] transferToPool #
       - modifiers: onlyController
    - [Pub] transferToUser #
       - modifiers: onlyController
    - [Ext] getStoredBalance
    - [Ext] batchTransferToPool #
       - modifiers: onlyController
    - [Ext] batchTransferToUser #
       - modifiers: onlyController
    - [Ext] farm #
       - modifiers: onlyFarmer
    - [Ext] setFarmer #
       - modifiers: onlyOwner

+  OtokenFactory (OtokenSpawner)
    - [Pub] <Constructor> #
    - [Ext] createOtoken #
    - [Ext] getOtokensLength
    - [Ext] getOtoken
    - [Ext] getTargetOtokenAddress
```

```
    - [Int] _getOptionId

+  OtokenSpawner
    - [Int] _spawn #
    - [Int] _computeAddress

+ [Lib] Actions
    - [Int] _parseOpenVaultArgs
    - [Int] _parseMintArgs
    - [Int] _parseBurnArgs
    - [Int] _parseDepositArgs
    - [Int] _parseWithdrawArgs
    - [Int] _parseRedeemArgs
    - [Int] _parseSettleVaultArgs
    - [Int] _parseLiquidateArgs
    - [Int] _parseCallArgs


($) = payable function
# = non-constant function
```

License

This report falls under the terms described in the included LICENSE.