

SALUS SECURITY

JAN 2023



CODE SECURITY ASSESSMENT

BONSAI STRIKE

Overview

Project Summary

- Name: Bonsai Strike
- Version: v1
- Platform: BSC
- Language: Solidity
- Repository:
 - <https://github.com/pengpeng/bonsai/commits/54a99607db7aad4e62bc9350d4ef6e529852157b>
- Audit Scope: See [Appendix A](#)

Project Dashboard

Application Summary

Name	Bonsai Strike
Version	v3
Type	Solidity
Dates	Jan 11 2023
Logs	Dec 12 2022; Dec 20 2022; Jan 11 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	1
Total informational issues	6
TOTAL	8

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.

Content

Introduction	4
1.1 About SALUS	4
1.2 Audit Breakdown	4
1.3 Disclaimer	4
Findings	5
2.1 Summary of Findings	5
2.2 Notable Findings	6
1. Incorrect handling of premium decimal	6
2. Inconsistent Solidity versions	9
2.3 Informational Findings	10
3. Redundant code	10
4. Custom error could be used	16
5. Incorrect comment	16
6. Magic numbers are used	18
7. External call to an out of scope address	19
8. Centralization risk	20
Appendix	21
Appendix A - Files in Scope	21

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (<https://t.me/salusec>), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Incorrect handling of premium decimal	Medium	Business Logic	Resolved
2	Inconsistent Solidity versions	Low	Configuration	Acknowledged
3	Redundant code	Informational	Redundancy	Acknowledged
4	Custom error could be used	Informational	Auditing and Logging	Acknowledged
5	Incorrect comment	Informational	Documentation	Resolved
6	Magic numbers are used	Informational	Code Quality	Acknowledged
7	External call to an out of scope address	Informational	Undefined Behavior	Acknowledged
8	Centralization risk	Informational	Centralization	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Incorrect handling of premium decimal

Severity: Medium

Category: Business Logic

Target:

- cvol/contracts/core/OptionsPremiumPricerInStables.sol
- core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol

Description

cvol/contracts/core/OptionsPremiumPricerInStables.sol:L122-152

```
function _getPremium(
    uint256 st,
    uint256 sp,
    uint256 expiryTimestamp,
    uint256 assetPrice,
    uint256 assetDecimals,
    bool isPut
) internal view returns (uint256 premium) {
    require(
        expiryTimestamp > block.timestamp,
        "Expiry must be in the future!"
    );

    uint256 v;
    uint256 t;
    (sp, v, t) = blackScholesParams(sp, expiryTimestamp);

    (uint256 call, uint256 put) = quoteAll(t, v, sp, st);

    // Multiplier to convert oracle latestAnswer to 18 decimals
    uint256 assetOracleMultiplier = 10**uint256(18).sub(assetDecimals);

    // Make option premium denominated in the underlying
    // asset for call vaults and USDC for put vaults
    premium = isPut
        ? wdiv(put, assetPrice.mul(assetOracleMultiplier))
        : wdiv(call, assetPrice.mul(assetOracleMultiplier));

    // Convert to 18 decimals
    premium = premium.mul(assetOracleMultiplier);
}
```

core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol: L337-374

```
function commitAndClose() external nonReentrant {
    address oldOption = optionState.currentOption;

    VaultLifecycle.CloseParams memory closeParams =
VaultLifecycle.CloseParams({
    OTOKEN_FACTORY: OTOKEN_FACTORY,
    USDC: USDC,
    currentOption: oldOption,
    delay: DELAY,
    lastStrikeOverrideRound: lastStrikeOverrideRound,
    overriddenStrikePrice: overriddenStrikePrice,
    strikeSelection: strikeSelection,
    optionsPremiumPricer: optionsPremiumPricer,
    premiumDiscount: premiumDiscount
    });

    (address otokenAddress, uint256 premium, uint256 strikePrice,
uint256 delta) =
VaultLifecycle.commitAndClose(closeParams, vaultParams,
vaultState);

    emit NewOptionStrikeSelected(strikePrice, delta);

    ShareMath.assertUint104(premium);
    currentOtokenPremium = uint104(premium);
    optionState.nextOption = otokenAddress;

    uint256 nextOptionReady = block.timestamp.add(DELAY);
    require(nextOptionReady <= type(uint32).max, "Overflow
nextOptionReady");
    optionState.nextOptionReadyAt = uint32(nextOptionReady);

    _closeShort(oldOption);
}
```

The **premium** returned from **_getPremium()** is converted to 18 decimal points. However, the premium has not been scaled to the asset's decimal in **commitAndClose()**. As a result, when the asset does not have 18 decimal points, the **currentOtokenPremium** would be miscalculated.

Recommendation

Consider converting the premium to the asset's decimal in **_getPremium()**.

Status

This issue has been resolved by the team in commit [d151b4d](#). The premium is scaled to the asset's decimal in **commitAndClose()**.

core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol: L337-381

```
function commitAndClose() external nonReentrant {
    address oldOption = optionState.currentOption;

    VaultLifecycle.CloseParams memory closeParams =
        VaultLifecycle.CloseParams({
            OTOKEN_FACTORY: OTOKEN_FACTORY,
            USDC: USDC,
            currentOption: oldOption,
            delay: DELAY,
            lastStrikeOverrideRound: lastStrikeOverrideRound,
            overriddenStrikePrice: overriddenStrikePrice,
            strikeSelection: strikeSelection,
            optionsPremiumPricer: optionsPremiumPricer,
            premiumDiscount: premiumDiscount
        });

    (
        address otokenAddress,
        uint256 premium,
        uint256 strikePrice,
        uint256 delta
    ) = VaultLifecycle.commitAndClose(closeParams, vaultParams,
vaultState);

    emit NewOptionStrikeSelected(strikePrice, delta);

    ShareMath.assertUint104(premium);

    // scaling premium to follow asset decimals
    uint256 decimals = vaultParams.decimals;
    premium = decimals > 18
        ? premium.mul(10**(decimals.sub(18)))
        : premium.div(10**(uint256(18).sub(decimals)));

    currentOtokenPremium = uint104(premium);
    optionState.nextOption = otokenAddress;

    uint256 nextOptionReady = block.timestamp.add(DELAY);
    require(
        nextOptionReady <= type(uint32).max,
        "Overflow nextOptionReady"
    );
    optionState.nextOptionReadyAt = uint32(nextOptionReady);

    _closeShort(oldOption);
}
```

2. Inconsistent Solidity versions

Severity: Low

Category: Configuration

Target: All files

Description

Different Solidity versions are used throughout the project. For example: **core** uses 0.8.4, **cvol** uses 0.7.3, and **opyn** uses 0.6.10

Using different Solidity versions in one project makes it hard to build and test the project. Moreover, using outdated versions of Solidity prevents you from benefiting from bug fixes and newer security checks in the newer version.

Recommendation

It is recommended to use consistent Solidity versions throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

3. Redundant code

Severity: Informational

Category: Redundancy

Target:

- core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol
- core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol
- core/contracts/libraries/VaultLifecycle.sol
- cvol/contracts/core/ManualVolOracle.sol
- opyn/contracts/pricers/CompoundPricer.sol
- opyn/contracts/pricers/WstethPricer.sol
- opyn/contracts/pricers/YearnPricer.sol

Description

1. The following comments and codes about WETH in RibbonThetaVaultWithSwap and RibbonVault are not used.

core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol: L22

```
//import {IWETH} from "../../../../../interfaces/IWETH.sol";
```

core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol: L82-83

```
/// @notice WETH9 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2
//address public immutable WETH;
```

core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol: L129-140

```
constructor(
    // address _weth,
    address _usdc,
    address _oTokenFactory,
    address _gammaController,
    address _marginPool,
    address _swapContract
    //) RibbonVault(_weth, _usdc, _gammaController, _marginPool,
_swapContract) {
    ) RibbonVault(_usdc, _gammaController, _marginPool, _swapContract) {
        require(_oTokenFactory != address(0), "!_oTokenFactory");
        OTOKEN_FACTORY = _oTokenFactory;
    }
}
```

core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol: L152-170

```
constructor(
    // address _weth,
    address _usdc,
    address _gammaController,
    address _marginPool,
```

```

    address _swapContract
) {
    //require(_weth != address(0), "!_weth");
    require(_usdc != address(0), "!_usdc");
    require(_swapContract != address(0), "!_swapContract");
    require(_gammaController != address(0), "!_gammaController");
    require(_marginPool != address(0), "!_marginPool");

    // WETH = _weth;
    USDC = _usdc;
    GAMMA_CONTROLLER = _gammaController;
    MARGIN_POOL = _marginPool;
    SWAP_CONTRACT = _swapContract;
}

```

core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol: L298-308

```

// /**
//  * @notice Deposits ETH into the contract and mint vault shares.
//  Reverts if the asset is not WETH.
//  */
// function depositETH() external payable nonReentrant {
//     require(vaultParams.asset == WETH, "!WETH");
//     require(msg.value > 0, "!value");

//     _depositFor(msg.value, msg.sender);

//     IWETH(WETH).deposit{value: msg.value}();
// }

```

core/contracts/vaults/BaseVaultWithSwap/base/RibbonVault.sol: L643-648

```

// if (asset == WETH) {
//     IWETH(WETH).withdraw(amount);
//     (bool success, ) = recipient.call{value: amount}("");
//     require(success, "Transfer failed");
//     return;
// }

```

2. The following codes related to naked are not used.

opyn/contracts/core/Controller.sol: L111-115

```

/// @dev mapping to store cap amount for naked margin vault per options
collateral asset (scaled by collateral asset decimals)
mapping(address => uint256) internal nakedCap;

/// @dev mapping to store amount of naked margin vaults in pool
mapping(address => uint256) internal nakedPoolBalance;

```

opyn/contracts/core/Controller.sol: L212-213

```

/// @notice emits an event when naked cap is updated
event NakedCapUpdated(address indexed collateral, uint256 cap);

```

opyn/contracts/core/Controller.sol: L412-424

```

/**

```

```

    * @notice set cap amount for collateral asset used in naked margin
    * @dev can only be called by owner
    * @param _collateral collateral asset address
    * @param _cap cap amount, should be scaled by collateral asset decimals
    */
function setNakedCap(address _collateral, uint256 _cap) external
onlyOwner {
    require(_cap > 0, "C36");

    nakedCap[_collateral] = _cap;

    emit NakedCapUpdated(_collateral, _cap);
}

```

opyn/contracts/core/Controller.sol: L620-604

```

/**
 * @notice get cap amount for collateral asset
 * @param _asset collateral asset address
 * @return cap amount
 */
function getNakedCap(address _asset) external view returns (uint256) {
    return nakedCap[_asset];
}

/**
 * @notice get amount of collateral deposited in all naked margin vaults
 * @param _asset collateral asset address
 * @return naked pool balance
 */
function getNakedPoolBalance(address _asset) external view returns
(uint256) {
    return nakedPoolBalance[_asset];
}

```

opyn/contracts/core/MarginCalculator.sol: L308-356

```

/**
 * @notice return the collateral required for naked margin vault, in
collateral asset decimals
 * @dev _shortAmount, _strikePrice and _underlyingPrice should be scaled
by 1e8
 * @param _underlying underlying asset address
 * @param _strike strike asset address
 * @param _collateral collateral asset address
 * @param _shortAmount amount of short otoken
 * @param _strikePrice otoken strike price
 * @param _underlyingPrice otoken underlying price
 * @param _shortExpiryTimestamp otoken expiry timestamp
 * @param _collateralDecimals otoken collateral asset decimals
 * @param _isPut otoken type
 * @return collateral required for a naked margin vault, in collateral
asset decimals
 */

```

```

function getNakedMarginRequired(
    address _underlying,
    address _strike,
    address _collateral,
    uint256 _shortAmount,
    uint256 _strikePrice,
    uint256 _underlyingPrice,
    uint256 _shortExpiryTimestamp,
    uint256 _collateralDecimals,
    bool _isPut
) external view returns (uint256) {
    bytes32 productHash = _getProductHash(_underlying, _strike,
_collateral, _isPut);

    // scale short amount from 1e8 to 1e27 (oToken is always in 1e8)
    FPI.FixedPointInt memory shortAmount =
FPI.fromScaledUint(_shortAmount, BASE);
    // scale short strike from 1e8 to 1e27
    FPI.FixedPointInt memory shortStrike =
FPI.fromScaledUint(_strikePrice, BASE);
    // scale short underlying price from 1e8 to 1e27
    FPI.FixedPointInt memory shortUnderlyingPrice =
FPI.fromScaledUint(_underlyingPrice, BASE);

    // return required margin, scaled by collateral asset decimals,
explicitly rounded up
    return
        FPI.toScaledUint(
            _getNakedMarginRequired(
                productHash,
                shortAmount,
                shortUnderlyingPrice,
                shortStrike,
                _shortExpiryTimestamp,
                _isPut
            ),
            _collateralDecimals,
            false
        );
}

```

opyn/contracts/core/MarginCalculator.sol: L762-809

```

/**
 * @notice get required collateral for naked margin position
 * if put:
 * a = min(strike price, spot shock * underlying price)
 * b = max(strike price - spot shock * underlying price, 0)
 * marginRequired = ( option upper bound value * a + b ) * short amount
 * if call:
 * a = min(1, strike price / (underlying price / spot shock value))
 * b = max(1- (strike price / (underlying price / spot shock value)), 0)
 * marginRequired = (option upper bound value * a + b) * short amount
 * @param _productHash product hash

```

```

* @param _shortAmount short amount in vault, in FixedPointInt type
* @param _strikePrice strike price of short otoken, in FixedPointInt
type
* @param _underlyingPrice underlying price of short otoken underlying
asset, in FixedPointInt type
* @param _shortExpiryTimestamp short otoken expiry timestamp
* @param _isPut otoken type, true if put option, false for call option
* @return required margin for this naked vault, in FixedPointInt type
(scaled by 1e27)
*/
function _getNakedMarginRequired(
    bytes32 _productHash,
    FPI.FixedPointInt memory _shortAmount,
    FPI.FixedPointInt memory _underlyingPrice,
    FPI.FixedPointInt memory _strikePrice,
    uint256 _shortExpiryTimestamp,
    bool _isPut
) internal view returns (FPI.FixedPointInt memory) {
    // find option upper bound value
    FPI.FixedPointInt memory optionUpperBoundValue =
    _findUpperBoundValue(_productHash, _shortExpiryTimestamp);
    // convert spot shock value of this product to FixedPointInt
    (already scaled by 1e27)
    FPI.FixedPointInt memory spotShockValue =
    FPI.FixedPointInt(int256(spotShock[_productHash]));

    FPI.FixedPointInt memory a;
    FPI.FixedPointInt memory b;
    FPI.FixedPointInt memory marginRequired;

    if (_isPut) {
        a = FPI.min(_strikePrice, spotShockValue.mul(_underlyingPrice));
        b =
        FPI.max(_strikePrice.sub(spotShockValue.mul(_underlyingPrice)), ZERO);
        marginRequired =
        optionUpperBoundValue.mul(a).add(b).mul(_shortAmount);
    } else {
        FPI.FixedPointInt memory one = FPI.fromScaledUint(1e27,
        SCALING_FACTOR);
        a = FPI.min(one,
        _strikePrice.mul(spotShockValue).div(_underlyingPrice));
        b =
        FPI.max(one.sub(_strikePrice.mul(spotShockValue).div(_underlyingPrice)),
        ZERO);
        marginRequired =
        optionUpperBoundValue.mul(a).add(b).mul(_shortAmount);
    }

    return marginRequired;
}

```

3. The **vol()** function in the **ManualVolOracle.sol** file is not used.
cvol/contracts/core/ManualVolOracle.sol: L50-52

```

/**
 * @notice Returns the standard deviation of the base currency in 10**8
 i.e. 1*10**8 = 100%
 * @return standardDeviation is the standard deviation of the asset
 */
function vol(bytes32) public pure returns (uint256 standardDeviation) {
    return 0;
}

```

4. **CompoundPricer.sol**, **WstethPricer.sol**, and **YearnPricer.sol** are not used.

5. The actionTypes of **DepositLongOption**, **WithdrawLongOption**, **Liquidate**, and **Call** are not used. Therefore, the lines of **_depositLong()**, **_withdrawLong()**, **_liquidate()**, **_call()** are not used.

opyn/contracts/core/Controller.sol: L667-685

```

if (actionType == Actions.ActionType.DepositLongOption) {
    _depositLong(Actions._parseDepositArgs(action));
}

if (actionType == Actions.ActionType.WithdrawLongOption) {
    _withdrawLong(Actions._parseWithdrawArgs(action));
}

if (actionType == Actions.ActionType.Liquidate) {
    _liquidate(Actions._parseLiquidateArgs(action));
}

if (actionType == Actions.ActionType.Call) {
    _call(Actions._parseCallArgs(action));
}

```

Recommendation

Consider removing all redundant codes. Any unused imports, inherited contracts, functions, parameters, variables, modifiers, events, or return values should be removed (or used appropriately) before mainnet deployment. This will not only reduce gas costs but also improve the readability and maintainability of the code.

Status

This issue has been acknowledged by the team.

4. Custom error could be used

Severity: Informational

Category: Auditing and Logging

Target:

- core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol

Description

core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol: L138

```
require(_oTokenFactory != address(0), "!_oTokenFactory");
```

Starting from Solidity v0.8.4, there is a convenient and gas-efficient way to explain to users why an operation failed through the use of custom errors. It can reduce both gas usage and bytecode size.

Recommendation

Consider using a custom error to replace the revert message.

```
error NotOTokenFactory();

if (_oTokenFactory == address(0)) {
    revert NotOTokenFactory();
}
```

Status

This issue has been acknowledged by the team.

5. Incorrect comment

Severity: Informational

Category: Documentation

Target:

- core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol
- opyn/contracts/core/Controller.sol
- core/contracts/libraries/VaultLifecycle.sol

Description

1. core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol: L101

```
/**
 * @notice Initialization parameters for the vault.
```

```

* @param _owner is the owner of the vault with critical permissions
* @param _feeRecipient is the address to receive vault performance and
management fees
* @param _managementFee is the management fee pct.
* @param _performanceFee is the performance fee pct.
* @param _tokenName is the name of the token
* @param _tokenSymbol is the symbol of the token
* @param _optionsPremiumPricer is the address of the contract with the
black-scholes premium calculation logic
* @param _strikeSelection is the address of the contract with strike
selection logic
* @param _premiumDiscount is the vault's discount applied to the
premium
* @param _auctionDuration is the duration of the gnosis auction
*/
struct InitParams {...}

```

The comment for `_auctionDuration` says it is the duration of the **gnosis** auction, but it should be the duration of the **airswap** auction.

2. opyn/contracts/core/Controller.sol: L647

```

actions except Settle, Redeem, Liquidate and Call are Vault-updating
actinos

```

The word “**actinos**” is misspelled, and should be “**actions**” instead.

3. core/contracts/libraries/VaultLifecycle.sol: L541

```

// At the first round, currentBalance=0, pendingAmount>0

```

But the implementation is **currentBalance == pendingAmount** at the first round.

Recommendation

Ensure that the code is well commented, both with NatSpec and inline comments, for better readability and maintainability. The comments should accurately reflect what the corresponding code does. Stale comments should be removed. Discrepancies between code and comments should be addressed.

Status

The `_auctionDuration` comment issue is resolved since the team has clarified that airswap is not related to `_auctionDuration`. The comment issues in `Controller` and `VaultLifecycle` have been fixed in the commit: [75cd7ed](#).

6. Magic numbers are used

Severity: Informational

Category: Code Quality

Target:

- cvol/contracts/core/ManualVolOracle.sol

Description

cvol/contracts/core/ManualVolOracle.sol:L111-L129

```
function setAnnualizedVol(
    bytes32[] calldata optionIds,
    uint256[] calldata newAnnualizedVols
) external onlyAdmin {
    require(
        optionIds.length == newAnnualizedVols.length,
        "Input lengths mismatched"
    );

    for (uint256 i = 0; i < optionIds.length; i++) {
        bytes32 optionId = optionIds[i];
        uint256 newAnnualizedVol = newAnnualizedVols[i];

        require(newAnnualizedVol > 50 * 10**6, "Cannot be less than
50%");
        require(newAnnualizedVol < 400 * 10**6, "Cannot be more than
400%");

        annualizedVols[optionId] = newAnnualizedVol;
    }
}
```

Magic numbers 50 and 400 are used in **setAnnualizedVol()**, which unnecessarily leads to potential error if the constants are changed during development.

Recommendation

Consider using constant variables to replace magic numbers, as this would make the code more maintainable and readable while costing nothing gas-wise (constants are replaced by their value at compile-time).

Status

This issue has been acknowledged by the team.

7. External call to an out of scope address

Severity: Informational

Category: Undefined Behavior

Target:

- core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol

Description

core/contracts/vaults/BaseVaultWithSwap/RibbonThetaVaultWithSwap.sol:L320-L331

```
function stake(uint256 numShares) external nonReentrant {  
    address _liquidityGauge = liquidityGauge;  
    require(_liquidityGauge != address(0)); // Removed revert msg due to contract size limit  
    require(numShares > 0);  
    uint256 heldByAccount = balanceOf(msg.sender);  
    if (heldByAccount < numShares) {  
        _redeem(numShares.sub(heldByAccount), false);  
    }  
    _transfer(msg.sender, address(this), numShares);  
    _approve(address(this), _liquidityGauge, numShares);  
    ILiquidityGauge(_liquidityGauge).deposit(numShares, msg.sender,  
false);  
}
```

There is a **stake()** function in the **RibbonThetaVaultWithSwap** contract, which invokes the **deposit()** function of **liquidityGauge**. However, the code in **liquidityGauge** address is out of the audit scope and unknown to the auditors. Therefore, there may be potential risks when making an external call to **liquidityGauge**.

Recommendation

Consider looking for an additional audit for the implementation of **liquidityGauge**. Also, consider disabling the **stake()** function for now.

Status

This issue has been acknowledged by the team. The team has clarified that the **liquidityGauge** is uninitialized and is address(0) by default, as a result, the **stake()** function is disabled due to the **require(_liquidityGauge != address(0))** check.

8. Centralization risk

Severity: Informational

Category: Centralization

Target:

- All files

Description

The addresses **Admin**, **Owner**, **keeper**, and **feeRecipient**, which have powerful control over the project, are all EOA addresses. If the private keys are compromised, the safety of the project will be seriously affected.

Recommendation

Consider using multi-signature contract wallets.

Status

This issue has been acknowledged by the team. The team has promised to use multi-signature wallets to mitigate centralization risk.

Appendix

Appendix A - Files in Scope

This audit covered the following files in commit [54a9960](#):

File	SHA-1 hash
GammaInterface.sol	f32c09383fd3ec77d360bb427d79cc72a2d467fe
ICRV.sol	5cc31baeacc4de6cb0e275f36d7e6ddd35dce69e
IERC20Detailed.sol	e5f61851c6aa3bd023189e9cc312cc88d25ed1e9
IGnosisAuction.sol	6f0631a6ad8a1ae892a4dcb0969e1594e5f7ffb6
ILiquidityGauge.sol	f3984ef60b8607cad6aa71961e656f2d52b5f100
IOptionsPurchaseQueue.sol	71c1e50ff77df564318789a320161a2bbd910d87
IPriceOracle.sol	d6ba148439631391d335e34f27134b6b654c41b9
IRibbon.sol	a16302ba9fe6657e80d528dac0ceef3babc26da
IRibbonThetaVault.sol	655b30b251e2dacd8431a5f8fc3672c6e64f0f50
ISAVAX.sol	c78cea544466ea8880d36f91d0a1fc0ce1992227
ISTETH.sol	ace28c090a6d938786aaa9216b8409e794cd9269
ISwap.sol	9d36bfa1d2a4f6ba142cd51b219dd8b71c2cab57
ISwapRouter.sol	0735ae72b2c8875e1166ecb3a21fc8994987a478
IUniswapV3Factory.sol	9e9141482c756853026389204b9f2b85548b50d8
IWETH.sol	a7c4d981e34291a6bedabb004b59f4625b921ff8
IYearn.sol	b097c13fa636af0cfc9e30af85f3d42c1f1ab96
BytesLib.sol	800b3ad6a47a50ea744d66832dc93d5c2a96d06b
DateTime.sol	2d2fcf86618bd5183e3a71bb8db656180f718fbd
GnosisAuction.sol	98e6f6a5d852dc57abb44b8d3aa0a67b1f34d7c4
Path.sol	39643ec19403274becfcac3b638280438fe7a70a
ShareMath.sol	8dc9fec357283feadf168029dff484759b21ae6
SupportsNonCompliantERC20.sol	49ac7f94d2732b0649837ea86fb73790fce4dbf1
UniswapRouter.sol	d8921a9594d54ab00cab25a12a4c772043037674

Vault.sol	f59896b5ddaca13bcbb79d88188e437146eb27a6
VaultLifecycle.sol	e267f5ebcddc30a8703ff62ce32761c90df5f970
OrynOracle.sol	a10bd007422ecbdeab2906508d59253b496a8702
RibbonThetaVaultStorage.sol	409dee91fc5bbd349054e658a3fe8c4a2b226268
ForceSend.sol	3104a5cb114a90ce97622fb264603d61042e0320
IAAssets.sol	754a15118983d0d3dd06540ebefa7847fdec555c
MockERC20.sol	f49ca9b9b2c19e52fae3373215002ac57d79c896
MockLiquidityGauge.sol	01e65cf5400ebc14e78a7899052a3f75c8dced7f
MockOptionsPremiumPricer.sol	2b5382b064eeb93f8b9e95fa2f86d51d971593a5
MockPriceOracle.sol	b89583eb2a0363a8cb1ce429bf71a4decc5256c3
MockRibbonVault.sol	75bb4f6a7a0c505ea091a81838025e2780764268
MockStrikeSelection.sol	f128111235017274e5ad1f0e5f42c72630a3d7b8
MockVolatilityOracle.sol	eb176641af9c3af42fa0c74c04b0501f83a90e05
TestShareMath.sol	7ec9a8038384a2e9e49bc55c440ec04a266af978
TestVaultLifecycle.sol	541c91100b6d2d0979e5a0703c40e93f2e8c06a4
DeltaStrikeSelection.sol	7516b8ae177400d79b02d32be98a1628e16c2781
OptionsPurchaseQueue.sol	a777a711fb4e033bdc55956ffb3864e91af2c044
PercentStrikeSelection.sol	134badb43cbd9c7664f1bce6e6180e579c49d72e
SAVAXDepositHelper.sol	e0d4c95d4e1f7b13f73573147126599a7b881bec
Swap.sol	77f0b342647becca0a7f55543f8fd4a7c514e91b
RibbonThetaVaultWithSwap.sol	bf566812a07fdb19fa4d7531d77941939b64191
RibbonVault.sol	ca756c3260a362124721e7a1b88ac7e4973bb7fc
CustomSafeERC20.sol	e825e5c7d22d882f2d3f0164d49f9fca4fbe2d2b
DSMath.sol	ffb2df8b224496f2c56d32f7010d95baffd388cd
AdminUpgradeabilityProxy.sol	5f775fdef4dd7e74e01d0280678d22d8d6a4511d
Proxy.sol	cff25ce2ec85ef7c3e48978d57b2b4c1d0048526
UpgradeabilityProxy.sol	b3e543d489f1459154eadd2ef4901985334f65b3
ChainlinkVolOracle.sol	1a527750502548191676b68e2285a3299fa3bc93
ManualVolOracle.sol	5296a5568af2effe5a7cf94a5c37cc16df952a0a

OptionsPremiumPricerInETH.sol	07c42b2feb267dc0a7cfcf3e680290d0e8b17dc5
OptionsPremiumPricerInStables.sol	1ccd61123db0cec0225781749bdbbb306d42a608
V3TwapVolOracle.sol	672fce6685f91b110bb239c1924a373942f8a32e
VolOracle.sol	480d3f79be49079c8b33f0afdf280c9d4fb4bc15
IERC20Detailed.sol	bcb6ea505a790343e7177125d1698445f332ffc
IManualVolatilityOracle.sol	7ff29a272d5a968b0a5bc845e74002ab1cb0ae28
IPriceOracle.sol	87e1e7edf39c18afe4247c936dd12539fa59a5a8
IVolatilityOracle.sol	5fe1907ca2fd2ee08c24d5f7734218d45c26c59d
DSMath.sol	9c4e9175b35bd4f52eb67200a8b3660db4e826f0
Math.sol	e235a8b5eaab3a160a55413b0e711565387ba0f0
OracleLibrary.sol	984640adafc3088d7891fb20c17bf355b275e306
PRBMathSD59x18.sol	76c3abe983e451ab40d69ac20cf8afc67070ea10
Welford.sol	221f6ac59535bd6d529976af0cf3e64de98f35c1
MockPriceOracle.sol	a46a5abee72126aa33b7fd5d78d273938f4137e5
TestMath.sol	d8b3ad33b1577e47919d5c1be8cf3e47dc9f85e9
TestMathV2.sol	913b9035042d9b98b15fe0cba788958906ce9867
TestOptionsPremiumPricerInETH.sol	d30bc5a628bd62f56c19d2800b29f64ef30125f7
TestOptionsPremiumPricerInStables.sol	7ab73ed3add9e9f7b5373105a03d01a26697ef80
TestVolOracle.sol	7654f7ffb5c0fc5d9bb316fa2f0695fba8ff6ca
TestWelford.sol	2d334e010d35f2ee41dff7add172b86abe5ac6eb
Migrations.sol	edd60454bd3b5d6055f126f7cde39d053dad63c9
AddressBook.sol	b99fbdf9f548e4ca5f04e155090011a7716e309a
Controller.sol	0fe1a56eb06204f6d89d4cb7a3bc3a1cd73d78e9
MarginCalculator.sol	3a4048d34b7a3cf47549e3e4d5b9712e23918f7f
MarginPool.sol	9f98e5150051badf0a36dd5bfd3cabdf3ecf801a
Oracle.sol	8a842959055b739aa8284fc7664909cb075b1e14
Otoken.sol	8e10c8069e4dc8dfeea9d58758bc72f29122b2be
OtokenFactory.sol	a941a43b147eaffa5a92e441d657a9e951d9da40

OtokenSpawner.sol	bab114a27fa6e70e63934c575675f417da9bbc8e
Whitelist.sol	814cb49b8520b3dae91f13d88e30f83f881ee945
PermitCallee.sol	b98ff2a706037baaa339e0f86e309798092ad81e
WETH9.sol	be3c1a5434b267aa5f51be9871368f3c2b53dd36
PayableProxyController.sol	b67b073b6afb961a5faab94c70b39cfa4cafbe92
AddressBookInterface.sol	518b3f743bda8aeb7942c54b279d381c45ab8ea1
AggregatorInterface.sol	dff31acdf964444f4554bc50eb3a717566cda95b
CalleeInterface.sol	e0dc8c0f0423110011ae5754601e68acd337f899
CTokenInterface.sol	66e81c6a9a366e74a7d7469d98a794e84ce712b6
ERC20Interface.sol	9ec6505dc3386fcca59fb07ffb1e7d3de3607716
MarginCalculatorInterface.sol	d41ff9d8f6a9d9e2e644671f514d632bd12bffe0
MarginPoolInterface.sol	634b55059efc8071c8419d4c8383a6f128728d63
OpynPricerInterface.sol	d941acf4aaa7b540778ac9c44efc25e85c27a940
OracleInterface.sol	72e147d7a5465cc162e8222508b37ee01a7cc33a
OtokenInterface.sol	a5a04f8d810cdea3c89a86ca795cb2406e6dba1d
WETH9Interface.sol	8c819b02fc1f3c09a3ece03c3b6b4722f76ced2b
WhitelistInterface.sol	f5a7c5f622d8f7883d4a0c44079c27c76929b3d7
WSTETHInterface.sol	9cffd80c4579e1e1cc9fdb4f24816342f4f32f5
YearnVaultInterface.sol	2c98cd74ebe0a2883a1c8acc6f122378a322086e
ZeroXExchangeInterface.sol	87a2836de9ef5248f491fe8b3bcdcf32962661b2
.gitkeep	da39a3ee5e6b4b0d3255bfef95601890afd80709
Actions.sol	4f5310b55174d5457790fe8347be0c1166db1008
FixedPointInt256.sol	3a3baf8e88dc1603906c2cd422ce83d907375ef27
MarginVault.sol	5f067b7b716b0645029104f6de80ba8dcd279418
SignedConverter.sol	050e0db5812019bc013ad943b4a1cde422a4c504
Mock0xERC20Proxy.sol	bb40c6db063eea81081bcca23a11dc4318d3ea22
Mock0xExchange.sol	3c6b94d464ad4efe0ff1e8b0413b6928415e04dc
MockAddressBook.sol	3b42547c1e7bae2bd42fc06a956bc416c323be61
MockChainlinkAggregator.sol	07f155389f071ace2e8e4bab1c73194f459e7241

MockController.sol	98f3e125824ab86a4cab459b09822f3d8714839b
MockCToken.sol	f10fbaa0b43b8c035de47cd2efa6586e35fd87cf
MockCUSDC.sol	e21ca00b540960e99f45323f73f22ae31ec4d4a4
MockDumbERC20.sol	8bfd5b67bf905d357faf5db1c9aa0d420acb564b
MockERC20.sol	66a36c09ed05c543bea8b1dd37d90cdb6bc34f53
MockOracle.sol	cd2f2c632e9d72e5d56511a56252f050d7251e6e
MockOtoken.sol	e2fbee6409b65dd2e5a2488996bc95bc58fcc7f9
MockPermitERC20.sol	fc071b4b1b4b08369c7bfd53a7038b70891813ce
MockPricer.sol	f789d5b27ed63aa7b1c623d3a7347764476b4f45
MockWhitelistModule.sol	8a1b71d10d33a2fb4fad3690c73a54b33ea446b1
MockWSTETHToken.sol	ca82b163ca3db3211e4e21819b4a9a30a0620f7e
MockYToken.sol	3227ce07e27043bd0e9868a50f48f11bccdfdc17
BokkyPooBahsDateTimeLibrary.sol	08d702fa1042386e32a1abff75775197552d8806
Spawn.sol	a000d2c0f5a077472916ec25f6a1f98a417a2eab
Address.sol	5c35eab0fd8ec53619773d46ce22834c47b591b5
Context.sol	4c4bd00502c02b85ea0398a0aeb939a523d5a140
Create2.sol	3a3457a29058359694415e56a1b4315ddaaec1f9
IERC20.sol	a6e30b43d308510593082253e7e7876f8e7695ac
Ownable.sol	d113729e806576bba2339e44a8389b414b790bd7
ReentrancyGuard.sol	fc497507ce5e4531bb8df1f19dfc50ffe27afbc9
SafeERC20.sol	37d216c05c752ae5d214f2873ca3183c309f9f7c
SafeMath.sol	f3306b3ebf161cdd355836f7d25836d8fb883ff6
SignedSafeMath.sol	9cf422aa8611e5326d681a979a0f0b6cdd084548
Strings.sol	af01547487871a00135caf318d3bee8dd3925b9f
ERC20Upgradeable.sol	0ccd3e65fd8bc09717d4f4d805d25a6870516aa0
IERC20Upgradeable.sol	1e132d7b45de6c9f51cad14053f8b5e8b42ecff7
Initializable.sol	68b9947efed22891fd56f4bd8732b8785a519d52
OwnableUpgradeSafe.sol	740417193350b0ff928c3c76eb5a90bbe33e0b51
OwnedUpgradeabilityProxy.sol	1a6e58fa15815599864c9f2e50f2f4898188e3a1

Proxy.sol	0f2752600b871b9b5870065a7f4c163fe9b3fafc
ReentrancyGuardUpgradeSafe.sol	1d284f1c7a81479467c5b5b93817b4f6c62a0af7
UpgradeabilityProxy.sol	e6026c4e7dcf0b41e09425f5b5787b57434564c8
ECDSAUpgradeable.sol	828338bdb740551fd4e6574e905dd7effaa8ecb2
EIP712Upgradeable.sol	d40d538e565a698eb18c96da5183c1016c956a69
ERC20PermitUpgradeable.sol	6e2bdd79d0afebcc76ba9f3c3654ab435da88e4d
IERC20PermitUpgradeable.sol	06d91147cc816cf42c6c9b44f111d67cc24e0d2e
ContextUpgradeable.sol	36eff65d137587233f2486ce4ee30028a4a56715
SafeMathUpgradeable.sol	4fd7f281bc36b610cdf9298ec653db7ddc47447c
CountersUpgradeable.sol	015156d86b8ac5163332717fa96ee27aac329dba
ChainlinkPricer.sol	453da1e6a7cc64e15ee297de3157db733c711389
CompoundPricer.sol	e77e141314085b34aadea0388206ee48ec13ce90
WstethPricer.sol	7d9d591bd9074c289fc049776f1dd4933d09d74a
YearnPricer.sol	21697f20f6f14b98a605801c8b93916e27a153f9
ActionTester.sol	0fc23e2239e35652cd6e4808673054e5c37e7c6c
CalculatorTester.sol	3dae851fe5454ddafceec917fe3cf2458a1891fc
CalleeAllowanceTester.sol	3c112ddbbaea63bc0761ba31c9423db070032e96
CallTester.sol	5ddff4a8e98b3cb9b9a4eaaabb20e2020de2b49c
FixedPointInt256Tester.sol	1c396b112b43e0d68febee4f110ef04d61a79f05
FlashUnwrap.sol	67d28931ce814b5b8131ff6a77bdd8f84ef8ed4f
MarginVaultTester.sol	0fc620fd002b61ee978120117d1bb7033b60a66a
OtokenImplV1.sol	f5677f970cad3390d8f815200f855eb0e375409
SignedConverterTester.sol	a18d05d77658727c03486f4eb51547c8fa8add42
UpgradeableContractV1.sol	b80b427eecd1fcea3a369e560fb0e3012e103195
UpgradeableContractV2.sol	a427615022e055be3caee953dba992792e28cfac