Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.05.15, the SlowMist security team received the Wombat Exchange team's security audit application for Wombat Exchange, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Audit Version

https://github.com/wombat-exchange/wombat (develop branch)

commit: 9ce1151a3a999d9c3d97e220f3a33e191cf7f876

Audit scope: (contains external dependencies)

- contracts/wombat-core/pool/CrossChainPool.sol
- contracts/wombat-core/pool/WormholeAdaptor.sol
- contracts/wombat-core/pool/PoolV3.sol
- contracts/wombat-core/pool/CoreV3.sol

- contracts/wombat-core/pool/VolatilePool.sol
- contracts/wombat-core/asset/ChainlinkPriceFeed.sol
- contracts/wombat-core/asset/PythPriceFeed.sol
- contracts/wombat-core/asset/OraclePriceFeed.sol

Fixed Version

https://github.com/wombat-exchange/wombat (develop branch)

commit: eb4890ce1015f3b920e7206bb7fe0dd5129a0bef

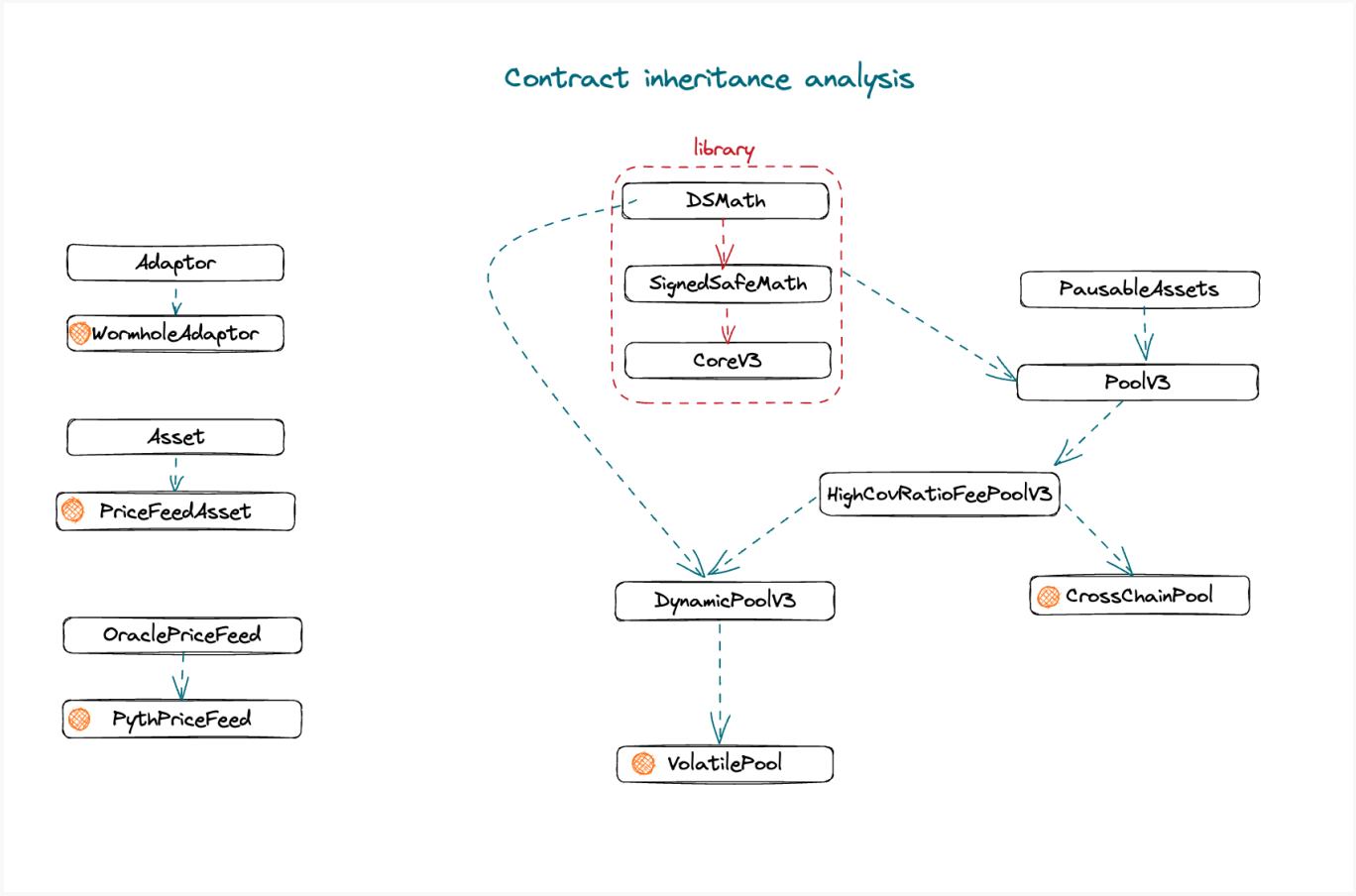
3.2 Vulnerability Information

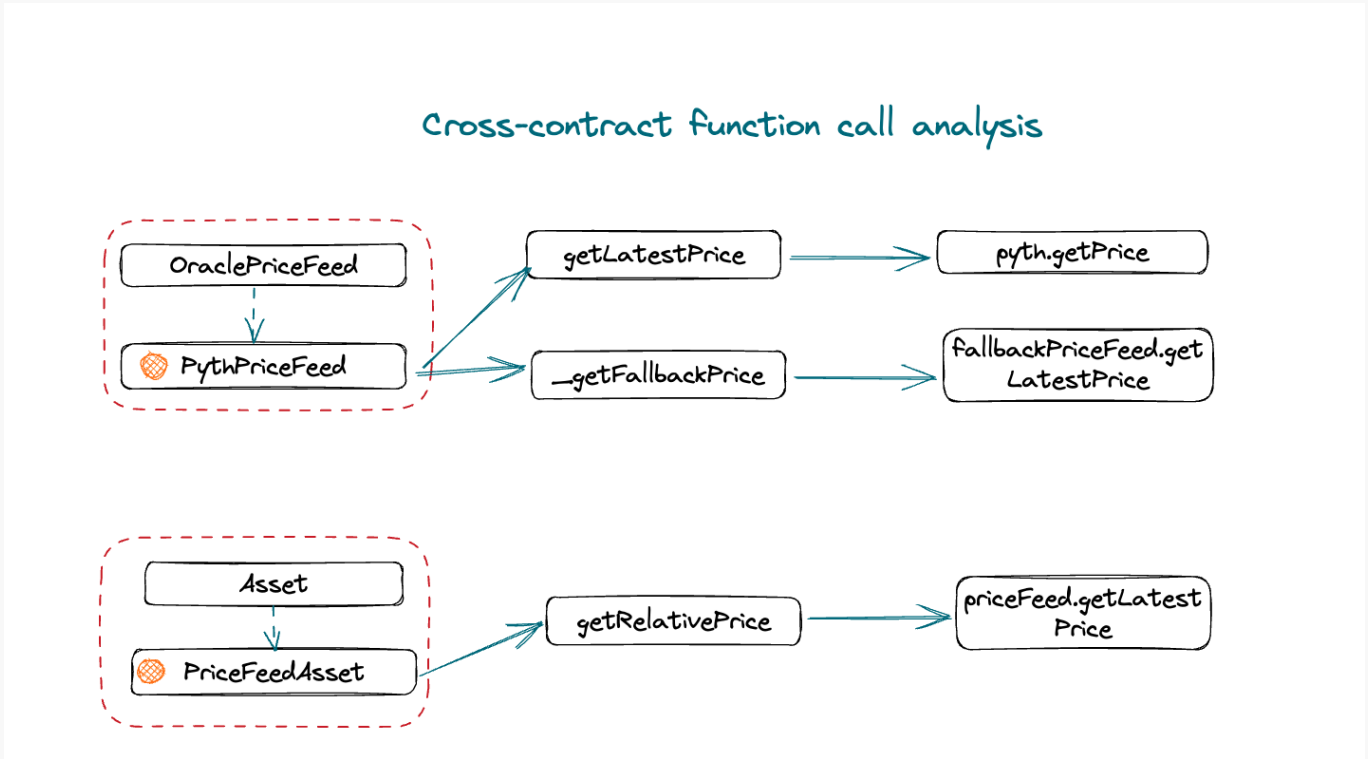
The following is the status of the vulnerabilities found in this audit:

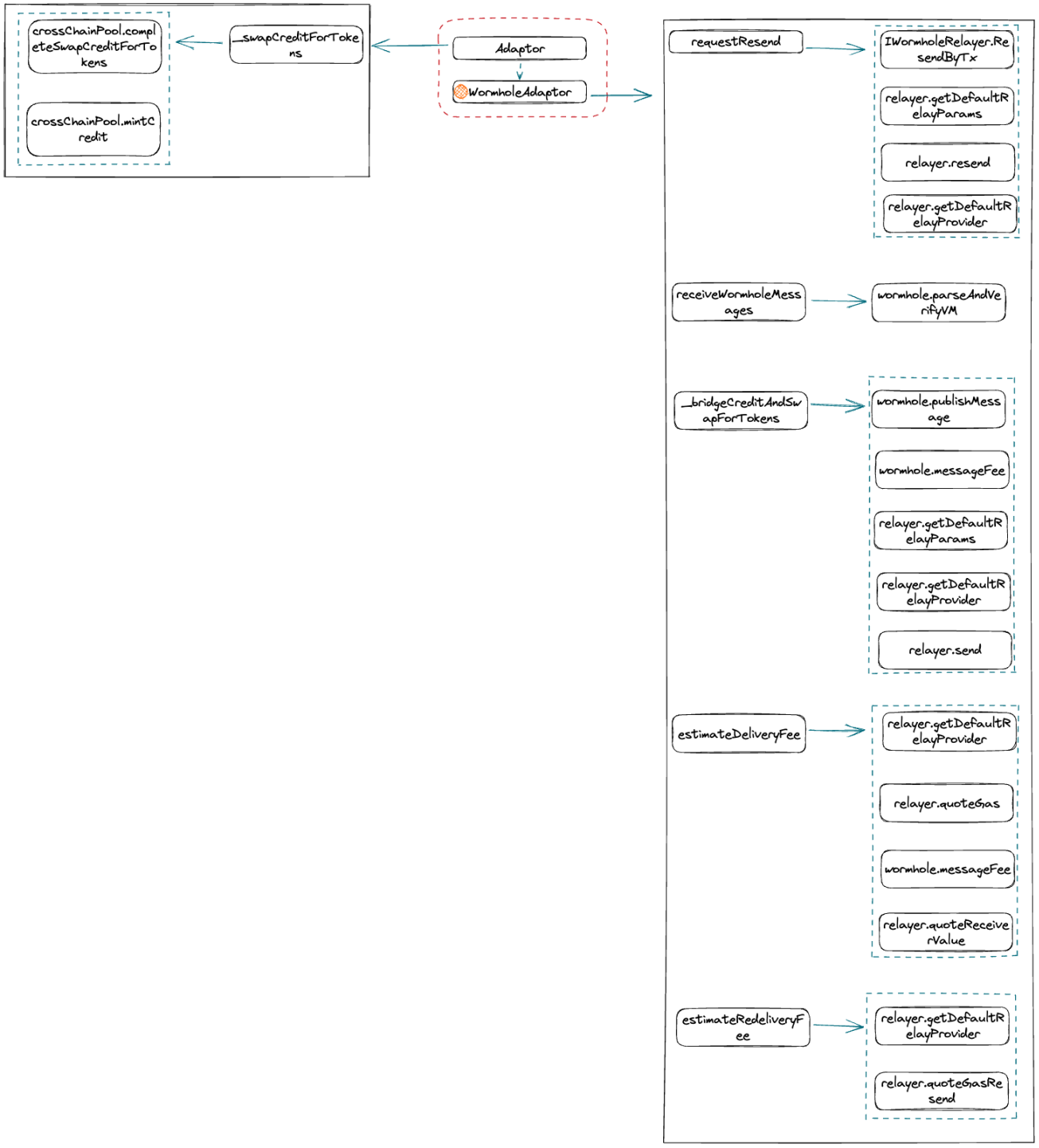
NO	Title	Category	Level	Status
N1	Overflow issues	Integer Overflow and Underflow Vulnerability	Critical	Fixed
N2	Missing event record	Others	Suggestion	Acknowledged
N3	Excessive authority issues	Authority Control Vulnerability Audit	Medium	Acknowledged
N4	Redundant judgment	Others	Suggestion	Acknowledged
N5	fee management suggestions	Authority Control Vulnerability Audit	Suggestion	Acknowledged
N6	conditional competition issues	Race Conditions Vulnerability	Low	Acknowledged
N7	Suggestions for variable type conversion	Others	Low	Fixed
N8	Business logic is unclear	Others	Suggestion	Acknowledged
N9	Token compatibility issues	Others	Suggestion	Acknowledged
N10	Redundant type conversion code	Others	Suggestion	Fixed

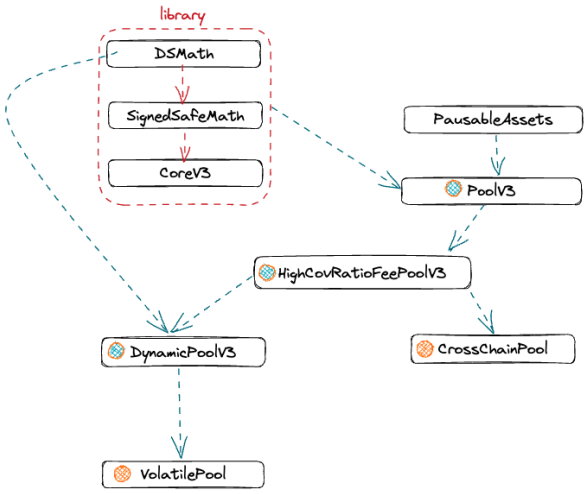
4 Code Overview

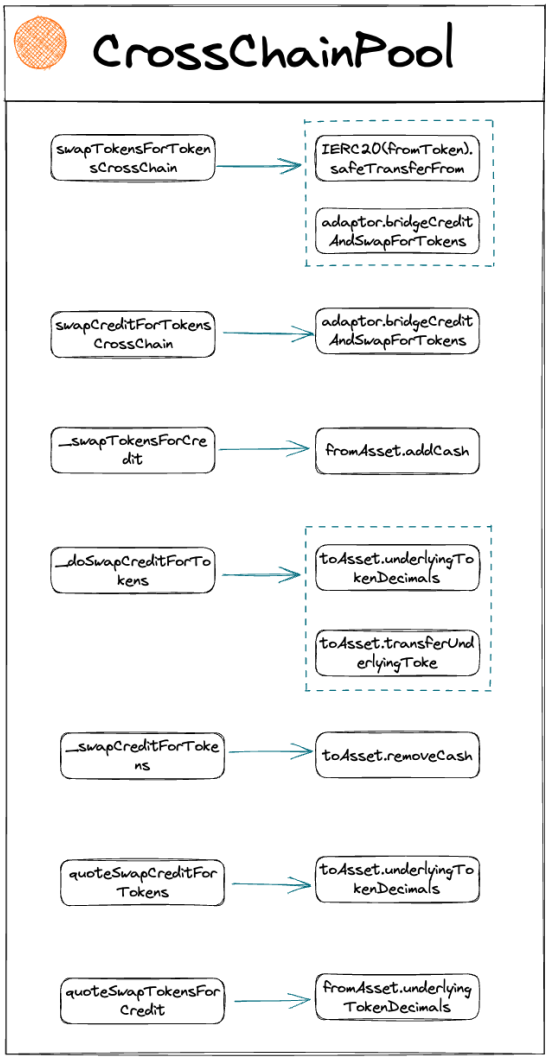
4.1 Contracts Description

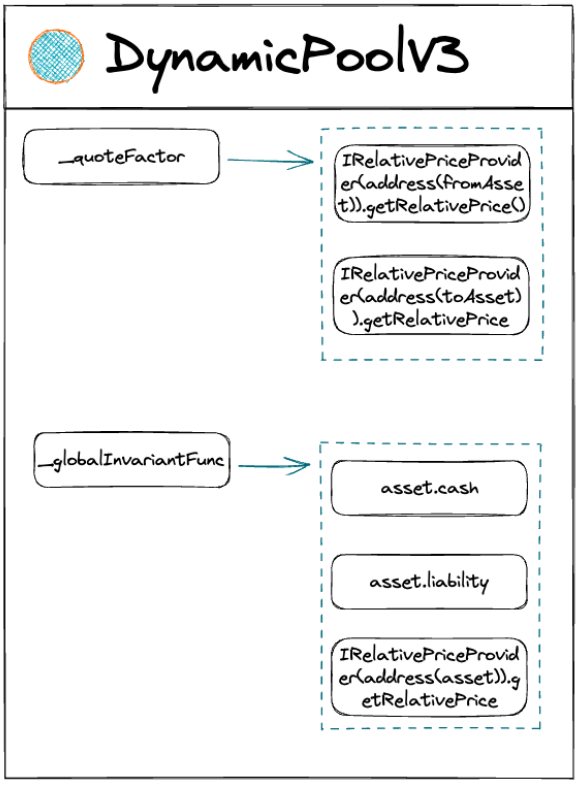


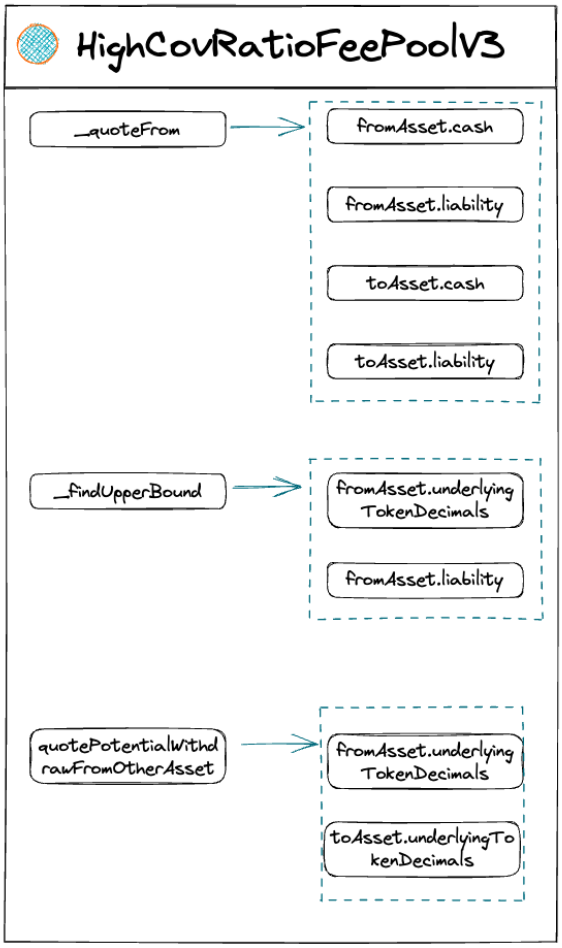


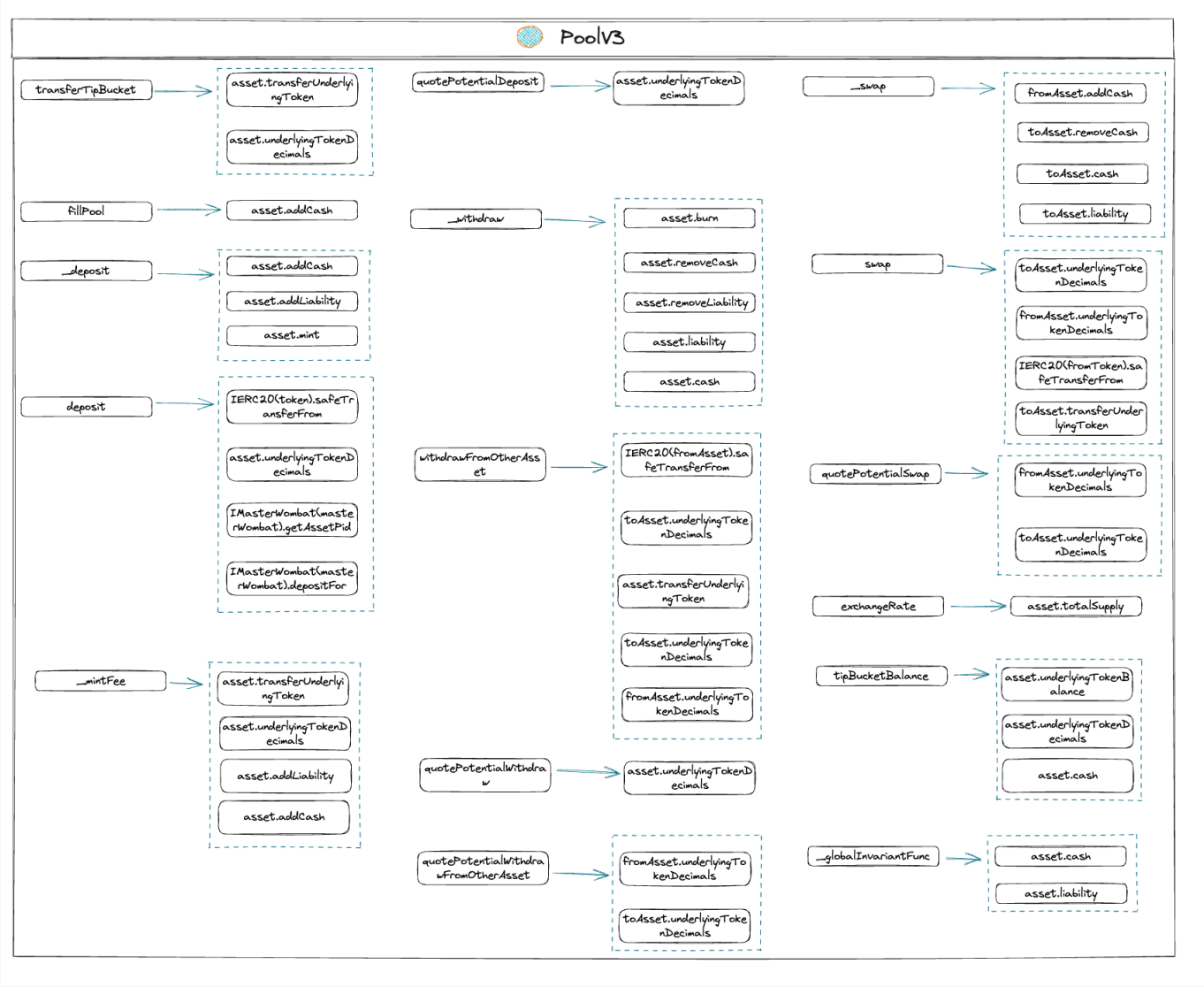

crossChainPool.completeSwapCreditForTokens
crossChainPool.mintCredit
_swapCreditForTokens
Adaptor
WormholeAdaptor
requestResend
IWormholeRelayer.resendByTx
relayer.getDefaultRelayParams
relayer.resend
relayer.getDefaultRelayProvider
receiveWormholeMessages
wormhole.parseAndVerifyVM
_bridgeCreditAndSwapForTokens
wormhole.publishMessage
wormhole.messageFee
relayer.getDefaultRelayParams
relayer.getDefaultRelayProvider
relayer.send
estimateDeliveryFee
relayer.getDefaultRelayProvider
relayer.quoteGas
wormhole.messageFee
relayer.quoteReceiverValue
estimateRedeliveryFee
relayer.getDefaultRelayProvider
relayer.quoteGasResend


library
DSMath
SignedSafeMath
CoreV3
PausableAssets
PoolV3
HighCovRatioFeePoolV3
DynamicPoolV3
CrossChainPool
VolatilePool


CrossChainPool
swapTokensForTokensCrossChain
IERC20(fromToken).safeTransferFrom
adaptor.bridgeCreditAndSwapForTokens
swapCreditForTokensCrossChain
adaptor.bridgeCreditAndSwapForTokens
_swapTokensForCredit
fromAsset.addCash
_doSwapCreditForTokens
toAsset.underlyingTokenDecimals
toAsset.transferUnderlyingToke
_swapCreditForTokens
toAsset.removeCash
quoteSwapCreditForTokens
toAsset.underlyingTokenDecimals
quoteSwapTokensForCredit
fromAsset.underlyingTokenDecimals


DynamicPoolV3
_quoteFactor
IRelativePriceProvider(address(fromAsset)).getRelativePrice()
IRelativePriceProvider(address(toAsset)).getRelativePrice
_globalInvariantFunc
asset.cash
asset.liability
IRelativePriceProvider(address(asset)).getRelativePrice


HighCovRatioFeePoolV3
_quoteFrom
fromAsset.cash
fromAsset.liability
toAsset.cash
toAsset.liability
_findUpperBound
fromAsset.underlyingTokenDecimals
fromAsset.liability
quotePotentialWithdrawFromOtherAsset
fromAsset.underlyingTokenDecimals
toAsset.underlyingTokenDecimals



The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

CoreV3			
Function Name	Visibility	Mutability	Modifiers
quoteDepositLiquidity	Public	-	-
quoteWithdrawAmount	Public	-	-
quoteWithdrawAmountFromOtherAsset	Public	-	-

CoreV3			
quoteSwap	Public	-	-
quoteSwapTokensForCredit	Public	-	-
quoteSwapCreditForTokens	Public	-	-
equilCovRatio	Public	-	-
swapQuoteFunc	Public	-	-
withdrawalAmountImpl	Public	-	-
withdrawalAmountInEquilImpl	Public	-	-
exactDepositLiquidityImpl	Public	-	-
exactDepositLiquidityInEquilImpl	Public	-	-
swapToCreditQuote	Public	-	-
swapFromCreditQuote	Public	-	-
highCovRatioFee	Public	-	-
_solveQuad	Internal	-	-
_invariantFunc	Internal	-	-
_coefficientFunc	Internal	-	-
_targetedCovRatio	Internal	-	-
_newEquilCovRatio	Internal	-	-
_newInvariantFunc	Internal	-	-
_highCovRatioFee	Internal	-	-

CrossChainPool			
Function Name	Visibility	Mutability	Modifiers
swapTokensForTokensCrossChai	External	Payable	nonReentrant

CrossChainPool			
n			whenNotPaused
swapCreditForTokens	External	Can Modify State	nonReentrant whenNotPaused
swapCreditForTokensCrossChain	External	Payable	nonReentrant whenNotPaused
_swapTokensForCredit	Internal	Can Modify State	-
_beforeSwapCreditForTokens	Internal	Can Modify State	-
_doSwapCreditForTokens	Internal	Can Modify State	-
_swapCreditForTokens	Internal	Can Modify State	-
quoteSwapCreditForTokens	External	-	-
quoteSwapTokensForCredit	External	-	-
globalEquilCovRatioWithCredit	External	-	-
_to128	Internal	-	-
completeSwapCreditForTokens	External	Can Modify State	whenNotPaused
mintCredit	External	Can Modify State	-
setSwapTokensForCreditEnabled	External	Can Modify State	onlyOwner
setSwapCreditForTokensEnabled	External	Can Modify State	onlyOwner
setMaximumOutboundCredit	External	Can Modify State	onlyOwner
setMaximumInboundCredit	External	Can Modify State	onlyOwner
setAdaptorAddr	External	Can Modify State	onlyOwner
setCrossChainHaircut	External	Can Modify State	onlyOwner

PoolV3			
Function Name	Visibility	Mutability	Modifiers
_checkLiquidity	Internal	-	-
_checkAddress	Internal	-	-
_checkSameAddress	Internal	-	-
_checkAmount	Internal	-	-
_ensure	Internal	-	-
_onlyDev	Internal	-	-
initialize	Public	Can Modify State	initializer
addAsset	External	Can Modify State	onlyOwner
removeAsset	External	Can Modify State	onlyOwner
setDev	External	Can Modify State	onlyOwner
setMasterWombat	External	Can Modify State	onlyOwner
setAmpFactor	External	Can Modify State	onlyOwner
setHaircutRate	External	Can Modify State	onlyOwner
setWithdrawalHaircutRate	External	Can Modify State	onlyOwner
setFee	External	Can Modify State	onlyOwner
transferTipBucket	External	Can Modify State	onlyOwner
setFeeTo	External	Can Modify State	onlyOwner
setMintFeeThreshold	External	Can Modify State	onlyOwner

PoolV3			
pause	External	Can Modify State	-
unpause	External	Can Modify State	-
pauseAsset	External	Can Modify State	-
unpauseAsset	External	Can Modify State	-
fillPool	External	Can Modify State	-
getTokens	External	-	-
_sizeOfAssetList	Internal	-	-
_getAsset	Internal	-	-
_getKeyAtIndex	Internal	-	-
_containsAsset	Internal	-	-
_assetOf	Internal	-	-
addressOfAsset	External	-	-
_deposit	Internal	Can Modify State	-
deposit	External	Can Modify State	nonReentrant whenNotPaused
quotePotentialDeposit	External	-	-
_withdraw	Internal	Can Modify State	-
withdraw	External	Can Modify State	nonReentrant whenNotPaused
withdrawFromOtherAsset	External	Can Modify State	nonReentrant whenNotPaused
quotePotentialWithdraw	External	-	-
quotePotentialWithdrawFromOtherAsset	External	-	-

PoolV3			
_quoteFactor	Internal	-	-
_quoteFrom	Internal	-	-
_swap	Internal	Can Modify State	-
swap	External	Can Modify State	nonReentrant whenNotPaused
quotePotentialSwap	Public	-	-
quoteAmountIn	External	-	-
exchangeRate	External	-	-
globalEquilCovRatio	Public	-	-
tipBucketBalance	Public	-	-
_globalInvariantFunc	Internal	-	-
_getGlobalEquilCovRatioForDepositWithdrawal	Internal	-	-
_mintFeeIfNeeded	Internal	Can Modify State	-
_mintFee	Internal	Can Modify State	-
_mintAllFees	Internal	Can Modify State	-
mintFee	External	Can Modify State	-

VolatilePool			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	-
setShouldCapEquilCovRatio	External	Can Modify State	onlyOwner
_getGlobalEquilCovRatioForDepositWithdraw	Internal	-	-

VolatilePool			
al			

WormholeAdaptor			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
requestResend	External	Payable	-
receiveWormholeMessages	External	Can Modify State	-
setAdaptorAddress	External	Can Modify State	onlyOwner
_recordMessageHash	Internal	Can Modify State	-
_bridgeCreditAndSwapForTokens	Internal	Can Modify State	-
estimateDeliveryFee	External	-	-
estimateRedeliveryFee	External	-	-
_wormholeAddrToEthAddr	Internal	-	-
_ethAddrToWormholeAddr	Internal	-	-

PriceFeedAsset			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Asset
setPriceFeed	External	Can Modify State	onlyOwner
getRelativePrice	External	-	-

PythPriceFeed			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer

PythPriceFeed			
getLatestPrice	External	-	-
setPriceID	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Critical] Overflow issues

Category: Integer Overflow and Underflow Vulnerability

Content

The data type of priceStruct.expo is int32, and the return result is negative, so uint256(int256(priceStruct.expo)) will get a large value, and `10 ** uint256(int256(priceStruct.expo)` will overflow. Because the `compiler version used is` pragma solidity ^0.8.5;`.

Code location: wombat/contracts/wombat-core/asset/PythPriceFeed.sol

```
function getLatestPrice(IERC20 _token) external view returns (uint256 price) {
    bytes32 priceID = priceIDs[_token];
    PythStructs.Price memory priceStruct = pyth.getPrice(priceID);

    // If the price is too old, use the fallback price feed
    if (block.timestamp - priceStruct.publishTime > maxPriceAge) {
        return _getFallbackPrice(_token);
    } else {
        require(priceStruct.price > 0);
        return (uint256(int256(priceStruct.price)) * 1e18) / 10 **
uint256(int256(priceStruct.expo));
    }
}
```

Solution

It is recommended to use priceStruct.price * 10 ** uint32(priceStruct.expo + 18))to extend the decimal to 1e18.

See: https://docs.pyth.network/pythnet-price-feeds/best-practices

Status

Fixed; This issue has been fixed in commit: 213f0aca3783b8af51080dfe182943eaf8f2b175.

[N2] [Suggestion] Missing event record

Category: Others

Content

The owner has the ability to modify the values of the maxPriceAge, shouldCapEquilCovRatio, startCovRatio, endCovRatio parameters, etc. which are global variables, but any modifications made to them are not recorded with events.

Code location: wombat/contracts/wombat-core/asset/OraclePriceFeed.sol

```
function setMaxPriceAge(uint96 _maxPriceAge) external onlyOwner {
        maxPriceAge = _maxPriceAge;
    }
```

Code location: wombat/contracts/wombat-core/pool/VolatilePool.sol

```
function setShouldCapEquilCovRatio(bool shouldCapEquilCovRatio_) external onlyOwner {
        shouldCapEquilCovRatio = shouldCapEquilCovRatio_;
    }
```

Code location: wombat/contracts/wombat-core/pool/HighCovRatioFeePoolV3.sol

```
function setCovRatioFeeParam(uint128 startCovRatio_, uint128 endCovRatio_) external
onlyOwner {
    if (startCovRatio_ < 1e18 || startCovRatio_ > endCovRatio_) revert
WOMBAT_INVALID_VALUE();

    startCovRatio = startCovRatio_;
    endCovRatio = endCovRatio_;
}
```

Code location: wombat/contracts/wombat-core/pool/CrossChainPool.sol

```
function setSwapTokensForCreditEnabled(bool enable) external onlyOwner {
    swapTokensForCreditEnabled = enable;
}

function setSwapCreditForTokensEnabled(bool enable) external onlyOwner {
    swapCreditForTokensEnabled = enable;
}
```

```
function setMaximumOutboundCredit(uint128 _maximumOutboundCredit) external onlyOwner
{
    maximumOutboundCredit = _maximumOutboundCredit;
}

function setMaximumInboundCredit(uint128 _maximumInboundCredit) external onlyOwner {
    maximumInboundCredit = _maximumInboundCredit;
}

function setAdaptorAddr(IAdaptor _adaptor) external onlyOwner {
    adaptor = _adaptor;
}

function setCrossChainHaircut(uint128 _tokensForCreditHaircut, uint128
_creditForTokensHaircut) external onlyOwner {
    require(_creditForTokensHaircut < 1e18 && _tokensForCreditHaircut < 1e18);
    creditForTokensHaircut = _creditForTokensHaircut;
    tokensForCreditHaircut = _tokensForCreditHaircut;
}
```

Code location: wombat/contracts/wombat-core/pool/HighCovRatioFeePoolV3.sol

```
function setCovRatioFeeParam(uint128 startCovRatio_, uint128 endCovRatio_) external
onlyOwner {
    if (startCovRatio_ < 1e18 || startCovRatio_ > endCovRatio_) revert
WOMBAT_INVALID_VALUE();

    startCovRatio = startCovRatio_;
    endCovRatio = endCovRatio_;
}
```

Code location: wombat/contracts/wombat-core/pool/WormholeAdaptor.sol

```
function setAdaptorAddress(uint16 wormholeChainId, address addr) external onlyOwner {
    adaptorAddress[wormholeChainId] = addr;
}
```

Solution

It is recommended to add event records for changes to global variables.

Status

Acknowledged; The project team response: due to the addition of event records, the size of the contract will become

larger, exceeding 24KB. If split the contract into multiple contracts by creating a library contract, it will affect the architecture of the code, so don't fix it for now.

[N3] [Medium] Excessive authority issues

Category: Authority Control Vulnerability Audit

Content

The Owner can modify priceIDs[_token], priceFeed, fallbackPriceFeed. This will affect the price at which the project gets oracle. The wrong price will lead to a fatal vulnerability in the project.

Code location: wombat/contracts/wombat-core/asset/PythPriceFeed.sol

```
function setPriceID(IERC20 _token, bytes32 _priceID) external onlyOwner {
    priceIDs[_token] = _priceID;
    emit UpdatepriceID(_token, _priceID);
}
```

Code location: wombat/contracts/wombat-core/asset/PriceFeedAsset.sol

```
function setPriceFeed(IPriceFeed _priceFeed) external onlyOwner {
    require(address(_priceFeed) != address(0), 'zero addr');
    priceFeed = _priceFeed;

    emit SetPriceFeed(_priceFeed);
}
```

Code location: wombat/contracts/wombat-core/asset/OraclePriceFeed.sol

```
function setFallbackPriceFeed(IPriceFeed _fallbackPriceFeed) external onlyOwner {
    fallbackPriceFeed = _fallbackPriceFeed;
    emit UpdateFallbackPriceFeed(_fallbackPriceFeed);
}
```

The owner can modify the configuration parameters in the contract. When the owner's permissions are used by hackers, it will affect the normal operation of the project.

Code location: wombat/contracts/wombat-core/pool/CrossChainPool.sol

```
function setSwapTokensForCreditEnabled(bool enable) external onlyOwner {
    swapTokensForCreditEnabled = enable;
}

function setSwapCreditForTokensEnabled(bool enable) external onlyOwner {
    swapCreditForTokensEnabled = enable;
}

function setMaximumOutboundCredit(uint128 _maximumOutboundCredit) external onlyOwner
{
    maximumOutboundCredit = _maximumOutboundCredit;
}

function setMaximumInboundCredit(uint128 _maximumInboundCredit) external onlyOwner {
    maximumInboundCredit = _maximumInboundCredit;
}

function setAdaptorAddr(IAdaptor _adaptor) external onlyOwner {
    adaptor = _adaptor;
}

function setCrossChainHaircut(uint128 _tokensForCreditHaircut, uint128
_creditForTokensHaircut) external onlyOwner {
    require(_creditForTokensHaircut < 1e18 && _tokensForCreditHaircut < 1e18);
    creditForTokensHaircut = _creditForTokensHaircut;
    tokensForCreditHaircut = _tokensForCreditHaircut;
}
```

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function addAsset(address token, address asset) external onlyOwner {
    _checkAddress(asset);
    _checkAddress(token);

    if (_containsAsset(token)) revert WOMBAT_ASSET_ALREADY_EXIST();
    _assets.values[token] = IAsset(asset);
    _assets.indexOf[token] = _assets.keys.length;
    _assets.keys.push(token);

    emit AssetAdded(token, asset);
}

/**
 * @notice Removes asset from asset struct
 * @dev Can only be called by owner
 * @param token The address of token to remove
```

```
 */
function removeAsset(address token) external onlyOwner {
    if (!_containsAsset(token)) revert WOMBAT_ASSET_NOT_EXISTS();

    address asset = address(_getAsset(token));
    delete _assets.values[token];

    uint256 index = _assets.indexOf[token];
    uint256 lastIndex = _assets.keys.length - 1;
    address lastKey = _assets.keys[lastIndex];

    _assets.indexOf[lastKey] = index;
    delete _assets.indexOf[token];

    _assets.keys[index] = lastKey;
    _assets.keys.pop();

    emit AssetRemoved(token, asset);
}

/**
 * @notice Changes the contract dev. Can only be set by the contract owner.
 * @param dev_ new contract dev address
 */
function setDev(address dev_) external onlyOwner {
    _checkAddress(dev_);
    dev = dev_;
    emit SetDev(dev_);
}

function setMasterWombat(address masterWombat_) external onlyOwner {
    _checkAddress(masterWombat_);
    masterWombat = masterWombat_;
    emit SetMasterWombat(masterWombat_);
}

/**
 * @notice Changes the pools amplification factor. Can only be set by the contract
owner.
 * @param ampFactor_ new pool's amplification factor
 */
function setAmpFactor(uint256 ampFactor_) external onlyOwner {
    if (ampFactor_ > WAD) revert WOMBAT_INVALID_VALUE(); // ampFactor_ should not be
set bigger than 1
    ampFactor = ampFactor_;
    emit SetAmpFactor(ampFactor_);
}

/**
```

```
     * @notice Changes the pools haircutRate. Can only be set by the contract owner.
     * @param haircutRate_ new pool's haircutRate_
     */
    function setHaircutRate(uint256 haircutRate_) external onlyOwner {
        if (haircutRate_ > WAD) revert WOMBAT_INVALID_VALUE(); // haircutRate_ should not
    be set bigger than 1
        haircutRate = haircutRate_;
        emit SetHaircutRate(haircutRate_);
    }

    function setWithdrawalHaircutRate(uint256 withdrawalHaircutRate_) external onlyOwner
    {
        if (withdrawalHaircutRate_ > WAD) revert WOMBAT_INVALID_VALUE();
        withdrawalHaircutRate = withdrawalHaircutRate_;
        emit SetWithdrawalHaircutRate(withdrawalHaircutRate_);
    }

    function setFee(uint256 lpDividendRatio_, uint256 retentionRatio_) external onlyOwner
    {
        if (retentionRatio_ + lpDividendRatio_ > WAD) revert WOMBAT_INVALID_VALUE();

        _mintAllFees();
        retentionRatio = retentionRatio_;
        lpDividendRatio = lpDividendRatio_;
        emit SetFee(lpDividendRatio_, retentionRatio_);
    }

    /**
     * @dev unit of amount should be in WAD
     */
    function transferTipBucket(address token, uint256 amount, address to) external
    onlyOwner {
        IAsset asset = _assetOf(token);
        uint256 tipBucketBal = tipBucketBalance(token);

        if (amount > tipBucketBal) {
            // revert if there's not enough amount in the tip bucket
            revert WOMBAT_INVALID_VALUE();
        }

        asset.transferUnderlyingToken(to,
    amount.fromWad(asset.underlyingTokenDecimals()));
        emit TransferTipBucket(token, amount, to);
    }

    /**
     * @notice Changes the fee beneficiary. Can only be set by the contract owner.
     * This value cannot be set to 0 to avoid unsettled fee.
     * @param feeTo_ new fee beneficiary
```

```
     */
    function setFeeTo(address feeTo_) external onlyOwner {
        _checkAddress(feeTo_);
        feeTo = feeTo_;
        emit SetFeeTo(feeTo_);
    }

    /**
     * @notice Set min fee to mint
     */
    function setMintFeeThreshold(uint256 mintFeeThreshold_) external onlyOwner {
        mintFeeThreshold = mintFeeThreshold_;
        emit SetMintFeeThreshold(mintFeeThreshold_);
    }
```

Solution

It is recommended that when updating the data of priceIDs[_token], priceFeed, and fallbackPriceFeed, the data should be strictly checked and the owner should use governance, timelock, or multi-sign contract for management.the owner should use governance, timelock, or multi-sign contract for management.

Status

Acknowledged; The project team response: Correct. We will use multisig owner.

[N4] [Suggestion] Redundant judgment

Category: Others

Content

If fromAmount is 0, the code will revert, so if (fromAmount >= 0) should be changed to if (fromAmount > 0).

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function quotePotentialSwap(
        address fromToken,
        address toToken,
        int256 fromAmount
    ) public view override returns (uint256 potentialOutcome, uint256 haircut) {
        _checkSameAddress(fromToken, toToken);
        if (fromAmount == 0) revert WOMBAT_ZERO_AMOUNT();

        IAsset fromAsset = _assetOf(fromToken);
        IAsset toAsset = _assetOf(toToken);

        fromAmount = fromAmount.toWad(fromAsset.underlyingTokenDecimals());
```

```
        (potentialOutcome, haircut) = _quoteFrom(fromAsset, toAsset, fromAmount);
        potentialOutcome =
potentialOutcome.fromWad(toAsset.underlyingTokenDecimals());
        if (fromAmount >= 0) {
            haircut = haircut.fromWad(toAsset.underlyingTokenDecimals());
        } else {
            haircut = haircut.fromWad(fromAsset.underlyingTokenDecimals());
        }
    }
```

Solution

It is recommended to change fromAmount >= 0 to fromAmount > 0.

Status

Acknowledged; The project team response: Acknowledge and won't fix. Even if we change the if branch to >0, the else branch is still technically <= 0.

[N5] [Suggestion] fee management suggestions

Category: Authority Control Vulnerability Audit

Content

If the receiving address of fee is an EOA address, there will be a single point risk of private key management.

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function setFeeTo(address feeTo_) external onlyOwner {
    _checkAddress(feeTo_);
    feeTo = feeTo_;
    emit SetFeeTo(feeTo_);
}
```

Solution

It is recommended to set the receiving address of the fee to multi-sign contracts.

Status

Acknowledged; The project team response: We use multisig as fee address.

[N6] [Low] conditional competition issues

Category: Race Conditions Vulnerability

Content

The fillPool function and the transferTipBucket function are controlled by the two roles of dev and owner respectively.

When the opinions of the dev and owner are inconsistent, there will be conditional competition issues.

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function fillPool(address token, uint256 amount) external {
    _onlyDev();
    IAsset asset = _assetOf(token);
    uint256 tipBucketBal = tipBucketBalance(token);

    if (amount > tipBucketBal) {
        // revert if there's not enough amount in the tip bucket
        revert WOMBAT_INVALID_VALUE();
    }

    asset.addCash(amount);
    emit FillPool(token, amount);
}
```

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function transferTipBucket(address token, uint256 amount, address to) external
onlyOwner {
    IAsset asset = _assetOf(token);
    uint256 tipBucketBal = tipBucketBalance(token);

    if (amount > tipBucketBal) {
        // revert if there's not enough amount in the tip bucket
        revert WOMBAT_INVALID_VALUE();
    }

    asset.transferUnderlyingToken(to,
amount.fromWad(asset.underlyingTokenDecimals()));
    emit TransferTipBucket(token, amount, to);
}
```

Solution

It is recommended to control the fillPool function and the transferTipBucket function by a single role.

Status

Acknowledged; The project team response: Acknowledge. Current dev and owner are the same multisig address. In our design, these two methods serve two different functionalities. Will revisit when we relax operations to governance.

[N7] [Low] Suggestions for variable type conversion

Category: Others

Content

The following functions when using uint256 to convert int256, it is not judged whether the variable to be converted is less than type(int256).max, and when using int256 to convert uint256, it is not judged whether the variable is greater than 0.

```
CoreV3.quoteDepositLiquidity
CoreV3.quoteWithdrawAmount
CoreV3.quoteWithdrawAmountFromOtherAsset
CoreV3.quoteSwap
CoreV3.quoteSwapTokensForCredit
CoreV3.quoteSwapCreditForTokens

PoolV3._globalInvariantFunc
PoolV3.globalEquilCovRatioWithCredit
DynamicPoolV3._globalInvariantFunc
PythPriceFeed.getLatestPrice
```

Solution

It is recommended that uint256 to convert int256 make sure the variable should be less than type(int256).max, and int256 to convert uint256 to make sure the variable should be greater than 0.

Status

Fixed; This issue has been fixed in commit: 412fcc40b8d739f7d6ab86277f655188903f1ecd. But there are still some global variables that haven't been fixed yet.

The project team response: they are intended since they are checked when configuring by the owner.

[N8] [Suggestion] Business logic is unclear

Category: Others

Content

The _quoteFactor function returns a fixed value of 1e18, but the function receives parameters, and the parameters do not need to be used.

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function _quoteFactor(
    IAsset, // fromAsset
    IAsset // toAsset
) internal view virtual returns (uint256) {
    return 1e18;
}
```

Solution

This seems to be the code to reserve features.

Status

Acknowledged; The project team response: The quoteFactor function is currently used by DynamicPool. We do not intend to use it for cross-chain purpose yet.

[N9] [Suggestion] Token compatibility issues

Category: Others

Content

When the project is transferred to the token, it does not judge the balance change before and after the transfer of the target address receiving the token, so it is incompatible with reflective tokens (deflation/inflation type tokens), which will cause the balance of the transfer to be inconsistent with the balance actually received, which will lead to calculation errors.

Code location: wombat/contracts/wombat-core/pool/PoolV3.sol

```
function deposit(
    address token,
    uint256 amount,
    uint256 minimumLiquidity,
    address to,
    uint256 deadline,
    bool shouldStake
) external override nonReentrant whenNotPaused returns (uint256 liquidity) {
    if (amount == 0) revert WOMBAT_ZERO_AMOUNT();
```

```
    _checkAddress(to);
    _ensure(deadline);
    requireAssetNotPaused(token);

    IAsset asset = _assetOf(token);
    IERC20(token).safeTransferFrom(address(msg.sender), address(asset), amount);

    if (!shouldStake) {
        liquidity = _deposit(asset, amount.toWad(asset.underlyingTokenDecimals()),
minimumLiquidity, to);
    } else {
        _checkAddress(masterWombat);
        // deposit and stake on behalf of the user
        liquidity = _deposit(asset, amount.toWad(asset.underlyingTokenDecimals()),
minimumLiquidity, address(this));

        asset.approve(masterWombat, liquidity);

        uint256 pid = IMasterWombat(masterWombat).getAssetPid(address(asset));
        IMasterWombat(masterWombat).depositFor(pid, liquidity, to);
    }

    emit Deposit(msg.sender, token, amount, liquidity, to);
}
```

Solution

At present, the design of the project is only compatible with the standard ERC20 Token. It is recommended to audit the compatibility of the token and the project when accessing the token to ensure that the project's funds will not be lost due to compatibility issues.

Status

Acknowledged; The project team response: Correct. By design, we will vet token before listing. We don't plan to support rebasing token that adjust on transfer.

[N10] [Suggestion] Redundant type conversion code

Category: Others

Content

The _quoteFrom function is using uint256 finalToAssetCovRatio = (toAssetCash +

uint256(actualToAmount)).wdiv(toAssetLiability); to convert uint256(actualToAmount), But actualToAmount is of the type uint256, it is no need to convert.

Code location: wombat/contracts/wombat-core/pool/HighCovRatioFeePoolV3.sol

```
function _quoteFrom(
        IAsset fromAsset,
        IAsset toAsset,
        int256 fromAmount
    ) internal view override returns (uint256 actualToAmount, uint256 haircut) {
        (actualToAmount, haircut) = super._quoteFrom(fromAsset, toAsset, fromAmount);

        if (fromAmount >= 0) {
            uint256 highCovRatioFee = CoreV3.highCovRatioFee(
                fromAsset.cash(),
                fromAsset.liability(),
                uint256(fromAmount),
                actualToAmount,
                startCovRatio,
                endCovRatio
            );

            actualToAmount -= highCovRatioFee;
            haircut += highCovRatioFee;
        } else {
            // reverse quote
            uint256 toAssetCash = toAsset.cash();
            uint256 toAssetLiability = toAsset.liability();
            uint256 finalToAssetCovRatio = (toAssetCash +
uint256(actualToAmount)).wdiv(toAssetLiability);
            if (finalToAssetCovRatio <= startCovRatio) {
                // happy path: no high cov ratio fee is charged
                return (actualToAmount, haircut);
            } else if (toAssetCash.wdiv(toAssetLiability) >= endCovRatio) {
                // the to-asset exceeds it's cov ratio limit, further swap to
increase cov ratio is impossible
                revert WOMBAT_COV_RATIO_LIMIT_EXCEEDED();
            }

            // reverse quote: cov ratio of the to-asset exceed endCovRatio. direct
reverse quote is not supported
            // we binary search for a upper bound
            actualToAmount = _findUpperBound(toAsset, fromAsset, uint256(-
fromAmount));
            (, haircut) = _quoteFrom(toAsset, fromAsset, actualToAmount.toInt256());
```

```
        }
    }
```

Solution

It is recommended to remove the conversion code.

Status

Fixed; This issue has been fixed in commit: eb4890ce1015f3b920e7206bb7fe0dd5129a0bef.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002305260001	SlowMist Security Team	2023.05.15 - 2023.05.26	Medium Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical risk, 1 medium risk, 1 low risk vulnerabilities and 7 suggestions. The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist