Custodia Security

Noya Mitigation Review

Conducted By: Ali Kalout, Ali Shehab

Contents

1. Disclaimer

A smart contract security review cannot ensure the absolute absence of vulnerabilities. This process is limited by time, resources, and expertise, aiming to identify as many vulnerabilities as possible. We cannot guarantee complete security after the review, nor can we assure that the review will detect every issue in your smart contracts. We strongly recommend follow-up security reviews, bug bounty programs, and on-chain monitoring.

2. Introduction

Custodia conducted a security assessment of Noya's smart contract following the resolution of issues identified in their Code4rena audit, ensuring the proper implementation of fixes.

3. About Noya

NOYA represents a paradigm shift in decentralized finance, introducing a protocol that empowers AI agents to control liquidity across multiple chains with unparalleled trustlessness and precision. Engineered with a foundational composable system, NOYA built from the ground up a secure private keeper network, a trustless AI-compatible oracle, and a competitive environment for AI architects alongside strategy managers.

4. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1. Impact

- High: Results in a substantial loss of assets within the protocol or significantly impacts a group of users.
- Medium: Causes a minor loss of funds (such as value leakage) or affects a core functionality of the protocol.
- Low: Leads to any unexpected behavior in some of the protocol's functionalities, but is not critical.

4.2. Likelihood

- High: The attack path is feasible with reasonable assumptions that replicate on-chain conditions, and the cost of the attack is relatively low compared to the potential funds that can be stolen or lost..
- Medium: The attack vector is conditionally incentivized but still relatively likely.
- Low: The attack requires too many or highly unlikely assumptions, or it demands a significant stake by the attacker with little or no incentive.

4.3. Action required for severity levels

- Critical: Must fix as soon as possible
- High: Must fix
- Medium: Should fix
- Low: Could fix

5. Security Assessment Summary

Repository: Noya-ai/noya-vault-contracts
Commit: 8279e96b8d276f52f96761c8d5ac173715da4e00

6. Executive Summary

Throughout the security review, Ali Kalout and Ali Shehab engaged with Noya to review Noya. In this period a total of Y issues were uncovered.

Findings Count

Severity	**Amount**
Critical	2
High	3
Medium	1
Low	1
Total Finding	**7**

Summary of C4 Fixes

ID	Title	Severity	Status
1426	`executeWithdraw` may be blocked if any of the users are blacklisted from the `baseToken`	High	Resolved
1224	`AccountingManager::resetMiddle` will not behave as expected	High	Resolved
1363	Loss of funds in `PendleConnector.depositIntoMarket()`	High	Resolved
1339	`PendleConnector` incorrectly sends the redeemed PT tokens to the market instead of the connect	High	Resolved
677	Decreasing a position in `PendleConnector` will remove it even if there's still a stake at Penpie	High	Resolved
350	Invalid calculation of position TVL in Pendle connector	High	Resolved
1438	Base tokens like USDT, USDC having different decimals on different chains can have their TVL updated incorrectly	High	Resolved
1430	`NoyaValueOracle.getValue` returns an incorrect price when a multi-token route is used	High	Resolved
1018	Invalid calculation of position TVL in Pendle connector	High	Resolved
991	PendleConnector.sol::supply doesn't pass a valid slippance protection min	High	Resolved
1033	BalancerConnector::_getPositionTVL is calculated incorrectly	High	Resolved
1021	BalancerConnector has incorrect implementation of totalSupply, positionTVL and total TVL will be invalid	High	Resolved
938	SiloConnector _getPositionTVL miscalculate the TVL position	High	Resolved

926	It is possible to open insolvent position is Silo connector, due to missing check in borrow function	High	Resolved
778	Numerous errors when calculating the TVL for the MorphoBlue connector	High	Resolved
708	`_getPositionTVL` of `UNIv3Connector` wrongly assumes ownership of all liquidity of the provided ticks inside `positionManager`	High	Resolved
1093	`Registry.sol#updateHoldingPosition` remove position logic is incorrect: should use ownerConnector instead of calculatorConnector when calculating `holdingPositionId`	High	Resolved
1522	AccountingManager contract's `previewDeposit`, `previewMint`, `previewWithdraw`, and `previewRedeem` functions are not compliant with EIP-4626 standard	Medium	Resolved
1334	AccountingManager has no correct implementations of the core ERC-4626 functions `deposit`, `mint`, `withdraw` and `redeem`	Medium	Resolved
1330	Attacker can increase the length of `withdrawQueue` by withdrawing 0 amount of tokens frequently	Medium	Resolved
1278	Withdrawals in AccountManager are prone to DOS attacks	Medium	Resolved
854	`depositQueue.queue` in `AccountingManager` can be flooded causing a DoS	Medium	Resolved
1097	`AccountingManager#totalWithdrawnAmount` should reflect tokens actually transferred to users, instead of expected transfers	Medium	Resolved
1329	`totalAssets()`, and thus `convertToShares()` and `convertToAssets()`, may revert, in violation of ERC-4626	Medium	Resolved
1488	Incorrect modifier condition	Medium	Resolved

1501	Stale price can be used in `getValueFromChainlinkFeed` function	Medium	Resolved
1415	Chainlink connector doesn’t check for the Min / Max prices returned	Medium	Resolved
917	Using the same heartbeat for multiple price feeds	Medium	Resolved
1428	`Keepers` does not implement EIP712 correctly on multiple occasions	Medium	Resolved
1298	The modifier `onlyExistingRoute` works incorrectly	Medium	Resolved
959	Dust donation might DOS all connectors to create new holding positions, by preventing removing existing holding positions	Medium	Resolved
799	The watchers cannot perform their role and can't do anything to intervene during bridging as stated by the docs	Medium	Resolved
1042	In the BalancerConnector, unclaimed rewards are not included in the calculation of the connectors TVL	Medium	Resolved
276	`lzSend()` forwards all of the contract balance as the native gas fee but the excess won't be always returned	Medium	Resolved
1321	Lack of function to claim reward in `AaveConnector`	Medium	Resolved
1554	Extra rewards are not updated in curve connector when harvestConvexRewards is called	Medium	Resolved
1110	`CurveConnector.sol#depositIntoConvexBooster` does not keep track of TVL if `stake == false`	Medium	Resolved
340	If a curve pool which CurveConnector uses is killed the vault manager can't close the position leading to loss of funds	Medium	Resolved
581	Some connectors prevents repayment of a borrow position if it doesn't leave the connector solvent or above minimumHealthFactor	Medium	Resolved

314	FullMath libabry is missing unchecked blocks, leading to DOS protocol's TVL and UniswapValueOracle	Medium	Resolved
830	MorphoBlueConnector:withdraw withdraws supplied tokens in a market order	Medium	Resolved

Summary of Findings

ID	Title	Severity	Status
[C-01]	Users can pass zero address to deposit, leading to deposited funds being stuck forever.	Critical	Resolved
[C-02]	`adjustIsolationModeAssetAsCollateral` and `changeEMode` are missing `onlyManager` modifier, allowing any user to call them	Critical	Resolved
[H-01]	`AccountingManager::executeWithdraw` is sending the wrong base token amount to `withdrawErrorsHandler`	High	Resolved
[H-02]	Invalid TVL calculation in MorphoBlueConnector::_getPositionTVL	High	Resolved
[H-03]	Invalid TVL calculation in BalancerConnector::_getPositionTVL	High	Resolved
[M-01]	`getValueFromChainlinkFeed` will result in stale prices	Medium	Resolved
[L-01]	Withdrawal errors are not cleared after being handled	Low	Resolved

7. Findings

7.1. Critical Findings

[C-01] Users can pass zero address to deposit, leading to deposited funds being stuck forever

Severity:

Critical

Description:

Users can call `AccountingManager::deposit` while passing a receiver address, this receiver address later gets minted some shares corresponding to the amount of tokens deposited. ERC4626/ERC20 reverts on minting to address 0, through the following:

```
/**
 * @dev Creates a `value` amount of tokens and assigns them to `account`, by transferring it
from address(0).
 * Relies on the `_update` mechanism
 *
 * Emits a {Transfer} event with `from` set to the zero address.
 *
 * NOTE: This function is not virtual, {_update} should be overridden instead.
 */
function _mint(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidReceiver(address(0));
    }
    _update(address(0), account, value);
}
```

At the same time, the protocol doesn't block users from calling deposit while passing `address(0)`, so later, when `executeDeposit` is called it'll always revert.

So all deposited funds of different users will get stuck forever.

Proof of Concept:

```
function test_DOSDepositQueue() public {
    _dealWhale(USDC, alice, USDC_Whale, 1_000_000e6);

    vm.startPrank(alice);
    SafeERC20.forceApprove(
        IERC20(USDC),
        address(accountingManager),
        type(uint256).max
    );

    accountingManager.deposit(address(0), 1e6, address(0));
    vm.stopPrank();

    vm.startPrank(owner);
    accountingManager.calculateDepositShares(10);

    vm.warp(block.timestamp + accountingManager.depositWaitingTime() + 1);

    vm.expectRevert(
        abi.encodeWithSelector(
            IERC20Errors.ERC20InvalidReceiver.selector,
            address(0)
        )
    );
    accountingManager.executeDeposit(10, connector, "");
    vm.stopPrank();
}
```

Recommendations:

Make sure to block users from passing the receiver (in `AccountingManager::deposit`) as zero address.

[C-02] `adjustIsolationModeAssetAsCollateral` and `changeEMode` are missing `onlyManager` modifier, allowing any user to call them

Severity:

Critical

Description:

Aave connector allows the protocol manager to interact with the Aave protocol to stake/borrow tokens, allowing them to earn yield. However, there are 2 functions `adjustIsolationModeAssetAsCollateral` and `changeEMode` that are permissionless. Allowing any user to change the connector's configuration on Aave, affecting its opened positions.

Proof of Concept:

```
function testPermissionlessAaveFunctions() public {
    address dummyUser = address(0x123);
    uint256 _amount = 100 * 1e6;
    _dealWhale(
        baseToken,
        address(connector),
        address(0x1AB4973a48dc892Cd9971ECE8e01DcC7688f8F23),
        _amount
    );

    vm.prank(owner);
    connector.supply(USDC, _amount);

    assertEq(IPool(aavePool).getUserEMode(address(connector)), 0);

    vm.prank(dummyUser);
    connector.changeEMode(1);

    assertEq(IPool(aavePool).getUserEMode(address(connector)), 1);

    vm.prank(dummyUser);
    connector.adjustIsolationModeAssetAsCollateral(USDC, false);

    vm.prank(owner);
    vm.expectRevert(bytes("34")); // 'The collateral balance is 0'
    connector.borrow(10e18, 2, DAI);
}
```

Recommendations:

Add `onlyManager` modifier to both `adjustIsolationModeAssetAsCollateral` and `changeEMode`.

7.2. High Findings

[H-01] `AccountingManager::executeWithdraw` is sending the wrong base token amount to `withdrawErrorsHandler`

Severity:
High

Description:
The protocol handles potential ERC20 transfer reverts by sending the amount to a `withdrawErrorsHandler` in `AccountingManager::executeWithdraw`. However, it is passing the wrong amount `data.amount` instead of `baseTokenAmount`, which is the "current" value while having fees subtracted from it.

This will drain the accounting manager, as more tokens than intended will be sent to the handler.

Proof of Concept:

```
function test_BlacklistReceiverWrongAmountSent() public {
    vm.prank(owner);
    accountingManager.setFees(5e4, 0, 0);

    _dealWhale(USDC, alice, USDC_Whale, 1000e6);

    RetrieveData[] memory retrieveData = new RetrieveData[](1);
    retrieveData[0] = RetrieveData(
        1e6,
        address(connector),
        abi.encode(1e6, hex"1232")
    );

    vm.startPrank(alice);
    SafeERC20.forceApprove(
        IERC20(USDC),
        address(accountingManager),
        type(uint256).max
    );
    accountingManager.deposit(alice, 100e6, address(0));
    vm.stopPrank();

    vm.startPrank(owner);
    accountingManager.calculateDepositShares(10);

    vm.warp(block.timestamp + accountingManager.depositWaitingTime() + 1);
    accountingManager.executeDeposit(10, connector, "");
    vm.stopPrank();

    vm.prank(Blacklist_ERC20(USDC).blacklister());
    Blacklist_ERC20(USDC).blacklist(alice);

    assertEq(Blacklist_ERC20(USDC).isBlacklisted(alice), true);

    vm.prank(alice);
    accountingManager.withdraw(1e6, alice);

    vm.startPrank(owner);
    accountingManager.calculateWithdrawShares(10);
    accountingManager.startCurrentWithdrawGroup();
    accountingManager.retrieveTokensForWithdraw(
        retrieveData,
        address(0),
        ""
    );
    accountingManager.fulfillCurrentWithdrawGroup();

    vm.warp(block.timestamp + accountingManager.withdrawWaitingTime() + 1);

    vm.expectRevert(); // ERC20: transfer amount exceeds balance
    accountingManager.executeWithdraw(10);
    vm.stopPrank();
}
```

Recommendations:

In `AccountingManager::executeWithdraw`, replace:

```
baseToken.safeTransfer(address(withdrawErrorsHandler), data.amount);
```

with:

```
baseToken.safeTransfer(address(withdrawErrorsHandler), baseTokenAmount);
```

[H-02] Invalid TVL calculation in `MorphoBlueConnector::_getPositionTVL`

Severity:

High

Description:

When calculating the TVL of a MorphoBlue position, the protocol manipulates the result of `convertCToL` to represent the answer in the loan token's decimals. However, this is wrong because MorphoBlue already handles this in https://github.com/morpho-org/morpho-blue/blob/main/src/interfaces/IOracle.sol.

This results in an inaccurate representation of the position's TVL.

Proof of Concept:

```
function testInvalidTVLCalculation() public {
    Id marketId = Id.wrap(
        0xb323495f7e4148be5643a4ea4a8221eef163e4bccfdedc2a6f4696baacbc86cc
    );

    uint256 USDCamount = 1_000e6;
    uint256 WETHamount = 1e18;

    _dealWhale(USDC, address(connector), USDC_Whale, USDCamount);
    _dealERC20(WSTETH, address(connector), WETHamount);

    assertEq(accountingManager.TVL(), 0);

    vm.prank(owner);
    connector.supply(WETHamount, marketId, false);

    assertEq(accountingManager.TVL(), 0);
}
```

Recommendations:

In `MorphoBlueConnector::_getPositionTVL`, replace:

```
supplyAmount - borrowAmount + (convertCToL(pos.collateral, params.oracle,
params.collateralToken) / 10**(collateralTokenDecimals) * 10**(loanTokenDecimals))
```

with:

```
(supplyAmount + convertCToL(pos.collateral, params.oracle, params.collateralToken)) -
borrowAmount
```

[H-03] Invalid TVL calculation in `BalancerConnector::_getPositionTVL`

Severity:

High

Description:

When calculating the TVL of a BalancerConnector position, the protocol wrongly computes the TVL of the Balancer position by doing a series of multiplications and divisions. The protocol should use the functions recommended in the Balancer docs, https://docs.balancer.fi/concepts/advanced/valuing-bpt/valuing-bpt.html#weighted-pools. This results in an inaccurate representation of the position's TVL.

Proof of Concept:

```
function testZeroPositionTVL() public {
    uint256[] memory amounts = new uint256[](4);
    uint256[] memory amountsW = new uint256[](3);
    uint256 USDCAmount = 10_000e6;
    _dealWhale(USDC, address(connector), USDC_Whale, USDCAmount);

    vm.startPrank(owner);

    addRoutesToNoyaOracle(address(USDT), address(USDC), address(840));

    connector.updateTokenInRegistry(USDC);

    assertEq(accountingManager.TVL(), USDCAmount);
    assertEq(IERC20(USDC).balanceOf(address(connector)), USDCAmount);

    amounts[2] = USDCAmount;
    amountsW[1] = USDCAmount;
    connector.openPosition(vanillaUsdcDaiUsdtId, amounts, amountsW, 0, 0);

    // TVL is 4449 wei
    // Connector holds 0 USDC
    // Connector holds 10k LP tokens - which should be translated to 10k USDC
    assertEq(accountingManager.TVL(), 4449);
    assertEq(IERC20(USDC).balanceOf(address(connector)), 0);
    assertGt(connector.totalLpBalanceOf(vanillaUsdcDaiUsdtId), 9900e18);
}
```

Recommendations:

In `BalancerConnector::_getPositionTVL`, replace the calculation logic with Balancer's recommendations, for example, for stable pools, it should be something similar to:

```
(address poolAddress, ) = IBalancerVault(balancerVault).getPool(
    pool.poolId
);
return
    ((lpBalance * IBalancerPool(poolAddress).getRate()) / 1e36) *
    10 ** IERC20Metadata(base).decimals();
```

7.3. MediumFindings

[M-01] `getValueFromChainlinkFeed` will result in stale prices

Severity:

Medium

Description:

According to the `updateChainlinkPriceAgeThreshold` function, the minimum possible chainlinkPriceAgeThreshold would be 1 hour. However, there are Chainlink oracles that have a heartbeat that is less than an hour; these oracles are essential for providing prices for the ERC20 tokens that should be supported by the protocol.
This was reported in the Code4rena contest https://github.com/code-423n4/2024-04-noya-findings/issues/1501, however, the fix still contains the same issue. As the introduced `updateChainlinkPriceAgeThreshold`, has the same "1 hour check", blocking maintainers from adding the correct heartbeat to each oracle.

```
function updateChainlinkPriceAgeThreshold(address source, uint256
_chainlinkPriceAgeThreshold)
    external
    onlyMaintainer
{
    if (_chainlinkPriceAgeThreshold <= 1 hours || _chainlinkPriceAgeThreshold >= 10 days) {
        revert NoyaChainlinkOracle_INVALID_INPUT();
    }
    chainlinkPriceAgeThreshold[source] = _chainlinkPriceAgeThreshold;
    emit ChainlinkPriceAgeThresholdUpdatedForAsset(source, _chainlinkPriceAgeThreshold);
}
```

Recommendations:

Refactor `updateChainlinkPriceAgeThreshold`'s condition to accommodate these oracles' heartbeats.

7.4. Low Findings

[L-01] Withdrawal errors are not cleared after being handled

Severity:

Low

Description:

Errors are not being cleared in `WithdrawErrorHandler::handleWithdrawalErrors`.

Recommendations:

Clear the error after handling it, add:

```
errors[errorId] = Error(address(0), address(0), 0, 0);
```