SHERLOCK

Security Review For Byzanlink

Collaborative Audit Prepared For: **Byzanlink**
Lead Security Expert(s): **pkqs90**
xiaoming90
Date Audited: **January 12 - January 13, 2026**

Introduction

Byzanlink V2 Vault is a non-custodial vault framework designed to support standardized on-chain asset management through the ERC-4626 tokenized vault standard, with ERC-2612 permit support. The system facilitates structured asset flows using both synchronous and asynchronous deposit and withdrawal mechanisms. The smart contracts operate with immutable logic and do not provide custodial control over user assets.

Scope

Repository: Byzanlink/ByzanlinkV2Vault

Audited Commit: 79b63d3e59a414615c3f8836cc279cfd3274a257

Final Commit: 16248d8536d3a69f8fe74a3c74350be015ba06ea

Files:

- contracts/adapters/pay5Adapter.sol

Final Commit Hash

16248d8536d3a69f8fe74a3c74350be015ba06ea

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	2	0

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue M-1: NAV inflation when operator calls allocate without new deposits [RESOLVED]

Source: https://github.com/sherlock-audit/2026-01-byzanlink-update-jan-12th/issues/2

Vulnerability Detail

The `Pay5Adapter` contract manages funds between a VaultV2 and a `borrowerVault`, tracking the total value via a `nav` (Net Asset Value) state variable. The `allocate()` function transfers USDC from the adapter to the `borrowerVault` and increases `nav` accordingly.

The `allocate()` function is callable by both the vault and the operator via the `onlyVaultOrOperator` modifier:

https://github.com/sherlock-audit/2026-01-byzanlink-update-jan-12th/blob/main/ByzanlinkV2Vault/contracts/adapters/pay5Adapter.sol#L139

```
function allocate(
    bytes memory /* data */,
    uint256 assets,
    bytes4 /* selector */,
    address /* sender */
)
    external
    override
    onlyVaultOrOperator
    nonReentrant
    returns (bytes32[] memory, int256 change)
{
    if (assets == 0) revert ErrorsLib.InvalidAmount();

    asset.safeTransfer(borrowerVault, assets);
    pay5State.nav += assets;
    ...
}
```

When called by the vault, new USDC is first transferred into the adapter, so the `nav` increase reflects actual new deposits. However, when called by the operator, no new USDC enters the system - the function simply moves existing USDC (e.g., funds that flowed back for pending withdrawals) to the `borrowerVault`. Despite this, `nav` is still incremented by `assets`.

Example: NAV = 10,000 USDC, and 500 USDC flows back to the adapter for withdrawal processing. If the operator calls `allocate(500)`, the 500 USDC moves back to `borrowerVault` and NAV increases to 10,500 USDC, even though total assets remain unchanged at 10,000 USDC.

Impact

NAV becomes inflated and out of sync with actual assets. Since `realAssets()` returns `pay5State.nav`, this inflated value affects share pricing calculations in VaultV2, potentially allowing users to redeem more than their fair share or causing accounting inconsistencies.

Recommendation

Only increment `nav` when the caller is the vault (indicating actual new deposits):

```
if (msg.sender == vault) {
    pay5State.nav += assets;
}
```

Issue M-2: `pay5State.nav` will be incorrect within the vault [RESOLVED]

Source: https://github.com/sherlock-audit/2026-01-byzanlink-update-jan-12th/issues/3

Vulnerability Detail

Assume the following sequence of actions:

1. `pay5State.nav` = 1000 USDC
2. `deallocateAsync()` is executed and an event is emitted
3. Trigger a offchain withdrawal request to withdraw 200 USDC from Credible.
4. 200 USDC sent to adaptor. Credible has 800 USDC now.
5. `updateNav()` is triggered. Since Credible only has 800 USDC, the new NAV will be 800.

From this point, two outcomes can happen:

1. Bound check can fail because 1000 to 800 is a big jump, and it will revert.
2. NAV gets updated to 800.

Assume that the NAV gets updated to 800. In this case, the `deallocate()` is triggered by the operator or `processWithdrawals()`, and `pay5State.nav -= 200`. So `pay5State.nav` becomes 600, which deviates from the Credible's total amount of 800.

Impact

The `pay5State.nav` will be incorrect in the vault.

Code Snippet

Tool Used

Manual Review

Recommendation

Discussed and confirmed resolved in final commit.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.