SHERLOCK

Security Review For GAIB

Collaborative Audit Prepared For:	GAIB
Lead Security Expert(s):	0x73696d616f antonttc
Date Audited:	March 28 - March 31, 2025
Final Commit:	142e3e8

Introduction

GAIB is building a yield-bearing token backed by real-world AI yield. This audit is focused on the “pre-deposit program” by boosting tvl from the community before our main protocol, an AI dollar called $AID, is released. By depositing into these pre-vaults, users could earn points, and we could use the idle funds to earn profits as future insurance pool of $AID.

Scope

Repository: gaib-ai/pre-vaults

Audited Commit: 7b0a81914a5116773b8b22f9a80dc0ddb8d63cda

Final Commit: 142e3e8687c4906a59f219e83854cb9600739a11

Files:

- contracts/pre-vaults/gpdUSDC.sol
- contracts/pre-vaults/gpdUSDS.sol
- contracts/pre-vaults/gpdUSDT.sol
- contracts/pre-vaults/gpdWBTC.sol
- contracts/pre-vaults/gpdWETH.sol
- contracts/token/utils/SafeMath.sol

Final Commit Hash

142e3e8687c4906a59f219e83854cb9600739a11

Findings

Each issue has an assigned severity:

- 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.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- 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
2	3	8

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue H-1: Wrong share to asset accounting during earn

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/18

Summary

The accounting system for yield-bearing assets is flawed, resulting in the loss of user funds or in some users receiving more than they should if they withdraw early.

Vulnerability Detail

Scenario one: Some depositors lose money

Admin turned on earn ==> interest accrued and increase `totalAsset` temporarily ==> use who deposit during this phase get diluted shares.

Scenario two: Some depositors get paid more

Some depositors deposit before `earn`, and got fix share => admin starts earn => totalAsset increase temporarily => user who withdraw now get their original deposit ”plus” the interest.

See the POC for both scenarios (user lose & user profit) here in this file

Impact

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e1066 99e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdUSDC.sol#L129

The culprit is using `totalAsset` as the token balance plus a token balance, when in reality the token changes should never be entitled to the users, as they are reserved for the protocol as a fee.

Tool Used

Manual Review

Recommendation

Simply change `totalAsset` as token balance + totalPrincipal

Issue H-2: `gpdWETH` double charges users who use `deposit` and `depositETH`

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/19

Summary

User who use `deposit` and `depositETH` got charged twice.

Found by @0xsimao

Vulnerability Detail

In the `deposit` and `depositETH` functions, an additional `transferFrom` (or depositing ETH into WETH) is called, even though it is already implemented in `super.deposit(assets, receiver);`.

You can find the POCs here:
https://github.com/sherlock-audit/2025-03-gaib/blob/97ffad567cedc131622bd2230d4fddf3d210b8fc/pre-vaults/test/pocs/High_WETH_Deposit.sol

Impact

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e106699e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdWETH.sol#L108

Tool Used

Manual Review

Recommendation

Remove the additional transferFrom, and manually call `mint()` directly for depositETH instead of using super.deposit.

Issue M-1: `PreDepositVault` will not work for USDT

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/8

Summary

`PreDepositVault::constructor()` uses `.approve()`, but this doesn't work for `USDT` as it does not return true.

Vulnerability Detail

USDT does not return on approval, which makes the EVM revert as it has return length checks and it expects it to return true.

Impact

Deployment of `PreDepositVault` for USDT fails.

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdUSDT.sol#L47

Tool Used

Manual Review

Recommendation

Use `.forceApprove()` from `SafeERC20`.

Issue M-2: `PreDepositVault` does not collect interest for `WBTC`

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/10

Summary

`WBTC` interest is disabled on Aave V3, as it is impossible to borrow.

Vulnerability Detail

See the market here for details https://app.aave.com/reserve-overview/?underlyingAsset=0x2260fac5e5542a773aa44fbcfedf7c193bc2c599marketName=proto_mainnet

Impact

No profit collection for WBTC

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdWBTC.sol#L12

Tool Used

Manual Review

Recommendation

Use an alternative yield source.

Issue M-3: No auto withdraw from yield protocols when liquidity is limited

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/16

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

No automatic withdrawal from yield protocols when liquidity is limited could result in a lock of funds.

Vulnerability Detail

In all vault contracts, there is no auto withdraw from underlying yield protocols during withdrawal. The admin can therefore effectively lock up users' funds by supplying all the liquidity to the yield source and earning interest on it.

Simple POC here

Impact

Temporary locking of funds, or the admin could maliciously lock up all assets.

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e106699e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdUSDC.sol#L110C4-L117C6

Tool Used

Manual Review

Recommendation

Add a simple try-catch logic that attempts to withdraw the difference amount from the underlying protocol if the vault has limited liquidity.

Issue L-1: `PreDepositVault::maxDeposit/Mint()` are missing `maxDepositLimit` as per the ERC4626 spec

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/9

Summary

`PreDepositVault` has a `maxDepositLimit` which is not reflected in `maxDeposit/Mint`, breaking the ERC4626 spec.

Vulnerability Detail

From the spec,

> MUST factor in both global and user-specific limits

Impact

Non EIP4626 compliance

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdUSDT.sol#L18

Tool Used

Manual Review

Recommendation

Override `ERC4626::maxDeposit/mint()` and implement the deposit limit there

Issue L-2: Use param instead of state variables to save gas in constructor

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/11

Summary

In the constructor, it is cheaper to use input parameters to avoid an additional SLOAD opcode in order to save gas.

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e106699e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdUSDC.sol#L42

Tool Used

Manual Review

Recommendation

Use

```
asset.approve(address(_aaveLendingPool), type(uint256).max);
```

Issue L-3: ATokens can be swept of the `PreDeposit Vault`

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/12

Summary

`PreDepositVault::sweep()` allows sweeping `aTokens`, decreasing users' deposits. Same for USDS.

Vulnerability Detail

`PreDepositVault::totalAssets()` is composed of `asset()` and `aToken`. Decreasing one of them leads to user losses.

Impact

User losses

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdWBTC.sol#L86

Tool Used

Manual Review

Recommendation

It's an admin function so could be left as is, but keep this in mind.

Issue L-4: Use of deprecated function

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/13

Summary

`aave.deposit` function on aave is deprecated, use supply instead

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e106699e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdUSDC.sol#L59

Tool Used

Manual Review

Recommendation

Use `supply` instead

Issue L-5: Rounding error in Aave Lending Pool

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/14

Summary

The aave lending pool produces rounding errors when calling `PreDepositVault::earn()`.

Vulnerability Detail

These errors were found to be negligible, being at most 1 wei, and they can either increase or decrease `totalAssets()` by 1. It shouldn't be exploitable but it's good to know.

Impact

`totalPrincipal` will be under/overestimated by 1 wei. For example, earning 1000 in `PreDepositVault::earn()`:

1. If it rounds up, `totalPrincipal` is 1000 but there will already be 1 profit, as we get 1001 aTokens
2. If it rounds down, `totalPrincipal` is 1000 but we only have 999 aTokens. Due to decimals in the assets, this isn't impactful. 1 wei is at most `1e-8` wbtc.

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdWBTC.sol#L56-L62

Tool Used

Manual Review

Recommendation

Rounding error is very small so no fix is needed.

Issue L-6: `PreDepositVault::sweep()` uses `address.transfer` which does not work for certain wallets

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/15

Summary

`address.transfer` forwards at most 2300 gas, which means it does not work for some smart contract wallets.

Vulnerability Detail

Admin calls `PreDepositVault::sweep()` using a smart contract wallet but it reverts trying to transfer native out.

Impact

Admin can't sweep ETH.

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/main/pre-vaults/contracts/pre-vaults/gpdWBTC.sol#L89

Tool Used

Manual Review

Recommendation

Use `sendValue()` https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol#L33.

Issue L-7: No event emitted for `setMaxDepositLimit`

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/20

Summary

No event emitted for setMaxDepositLimit

Code Snippet

https://github.com/sherlock-audit/2025-03-gaib/blob/7b0ededc742877fb6ceab1e106699e1abf1f98ad/pre-vaults/contracts/pre-vaults/gpdWBTC.sol#L96

Tool Used

Manual Review

Recommendation

Add events for better off-chain infrastructure handling.

Issue L-8: `sweep` function doesn't work with certain token

Source: https://github.com/sherlock-audit/2025-03-gaib/issues/21

Summary

Use transfer instead of safeTransfer in the sweep function to support as many tokens as possible.

Vulnerability Detail

The `sweep` function could make use of `IERC20::transfer` instead of safeTransfer to ensure maximal compatibility. Since this is a recovery function, the widest list of possible tokens should be supported.

For example, USDT on the Tron network is incompatible with safeTransfer as it always returns false.

Tool Used

Manual Review

Recommendation

Use `transfer` in rescue functions

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.