HALBORN

Affine DeFi - Multiplyr

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 19th, 2022 - October 7th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/04/2022	Omar Alshaeb
0.2	Draft Review	10/07/2022	Kubilay Onur Gungor
0.3	Draft Review	10/07/2022	Gabi Urrutia
1.0	Remediation Plan	10/26/2022	Omar Alshaeb
1.1	Remediation Plan Review	10/28/2022	Kubilay Onur Gungor
1.2	Remediation Plan Review	10/28/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Kubilay Onur Gungor	Halborn	Kubilay.Gungor@halborn.com

Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Affine DeFi engaged Halborn to conduct a security audit on their smart contracts beginning on September 19th, 2022 and ending on October 7th, 2022. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were mostly addressed by the Affine DeFi team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- src/ethereum/L1CompoundStrategy.sol
- src/ethereum/L1Vault.sol
- src/ethereum/L1WormholeRouter.sol
- src/external/Multicall.sol
- src/interfaces/*
- src/polygon/Detailed.sol
- src/polygon/EmergencyWithdrawalQueue.sol
- src/polygon/ERC4626Router.sol
- src/polygon/ERC4626RouterBase.sol
- src/polygon/Forwarder.sol
- src/polygon/L2AAVEStrategy.sol
- src/polygon/L2Vault.sol
- src/polygon/L2WormholeRouter.sol
- src/polygon/Router.sol
- src/polygon/TwoAssetBasket.sol
- src/AffineGovernable.sol
- src/BaseStrategy.sol
- src/BaseVault.sol
- src/BridgeEscrow.sol
- src/Constants.sol
- src/DollarMath.sol
- src/WormholeRouter.sol

Commit ID: 30e93568ca0b0b458f8744bae1e62aaf1e132647

And the following smart contracts:

- src/ethereum/CurveStrategy.sol
- src/ethereum/ConvexStrategy.sol
- src/polygon/DeltaNeutralLp.sol

Commit ID: 06d6bc37fa80f0fdf794a8cb93e8100288d065e0

Fixed Commit ID: 06d6bc37fa80f0fdf794a8cb93e8100288d065e0

And the following smart contracts:

- src/BaseVault.sol
- src/ethereum/L1Vault.sol
- src/polygon/L2Vault.sol

Commit ID: 302ab4e2e54c2666d607be1b88861636fdee311d

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	2	2

LIKELIHOOD

IMPACT

🟨 (HAL-01) (HAL-02)	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩 (HAL-03)	🟨	🟨	🟥
⬜	🟩 (HAL-04)	🟩	🟨	🟨
⬜ (HAL-05) (HAL-06)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - IGNORE EXTERNAL CALL FEE	Medium	SOLVED - 10/26/2022
HAL02 - POSSIBLE LOSS OF FUNDS	Medium	SOLVED - 10/26/2022
HAL03 - POSSIBLE UNPREDICTABILITY BETWEEN L2 AND L1 RATIOS	Low	RISK ACCEPTED
HAL04 - FUNCTION DOES NOT CHECK THE TOKEN BALANCE BEFORE AND AFTER A CALL	Low	SOLVED - 10/26/2022
HAL05 - LACK OF PROPER SLIPPAGE PROTECTION	Informational	SOLVED - 10/26/2022
HAL06 - POSSIBLE MISUSE OF CHAIN ID	Informational	SOLVED - 10/26/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) IGNORE EXTERNAL CALL FEE - MEDIUM

Description:

The wormhole publishMessage function is payable. Currently, requires no fees, but that can be changed over time. If the wormhole decides to set a fee greater than 0, all those external calls within the protocol would fail. Hence, leaving the wormhole routers unable to perform their critical tasks.

Code Location:

Listing 1: L2WormholeRouter.sol (Line 33)

```
function reportTransferredFund(uint256 amount) external {
    require(msg.sender == address(vault), "Only vault");
    bytes memory payload = abi.encode(Constants.
↳ L2_FUND_TRANSFER_REPORT, amount);
    uint64 sequence = wormhole.nextSequence(address(this));
    wormhole.publishMessage(uint32(sequence), payload,
↳ consistencyLevel);
}
```

Listing 2: L2WormholeRouter.sol (Line 40)

```
function requestFunds(uint256 amount) external {
    require(msg.sender == address(vault), "Only vault");
    bytes memory payload = abi.encode(Constants.L2_FUND_REQUEST,
↳ amount);
    uint64 sequence = wormhole.nextSequence(address(this));
    wormhole.publishMessage(uint32(sequence), payload,
↳ consistencyLevel);
}
```

Listing 3: L1WormholeRouter.sol (Line 37)

```
function reportTVL(uint256 tvl, bool received) external {
    require(msg.sender == address(vault), "Only vault");
    bytes memory payload = abi.encode(Constants.L1_TVL, tvl,
↳ received);
    // NOTE: We use the current tx count (to wormhole) of this
↳ contract
    // as a nonce when publishing messages
    // This casting is fine so long as we send less than 2 ** 32 -
↳  1 (~ 4 billion) messages
    uint64 sequence = wormhole.nextSequence(address(this));

    wormhole.publishMessage(uint32(sequence), payload,
↳ consistencyLevel);
}

```

Listing 4: L1WormholeRouter.sol (Line 45)

```
function reportTransferredFund(uint256 amount) external {
    require(msg.sender == address(vault), "Only vault");
    bytes memory payload = abi.encode(Constants.
↳ L1_FUND_TRANSFER_REPORT, amount);
    uint64 sequence = wormhole.nextSequence(address(this));

    wormhole.publishMessage(uint32(sequence), payload,
↳ consistencyLevel);
}

```

Proof of Concept:

1. Wormhole publishMessage function increase its fee transaction
2. Affine DeFi wormhole routers fail to publish messages due to not sending any fee on the transaction
3. Affine DeFi overall protocol does not properly work

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

Considering the need to call publishMessage, paying transaction fees is strongly recommended.

Remediation Plan:

SOLVED: The Affine DeFi team solved the issue in commit: 06d6bc37fa80f0fdf794a8cb93e8100288d065e0

3.2 (HAL-02) POSSIBLE LOSS OF FUNDS - MEDIUM

Description:

Wormhole does not fail if the destination chain ID is different from the one supposed to be. If the rebalancer bot calls this function directly with a different chain ID, it will not fail, so funds during the transactions can be lost.

You can check the Wormhole Chain IDs on each chain, which is not the same as the network chain ID and can be easily confused.

Code Location:

Listing 5: WormholeRouter.sol (Line 43)

```
41 function _validateWormholeMessageEmitter(IWormhole.VM memory vm) internal view {
42     require(vm.emitterAddress == bytes32(uint256(uint160(otherLayerRouter))), "Wrong emitter address");
43     require(vm.emitterChainId == otherLayerChainId, "Wrong emitter chain");
44     require(vm.nonce >= nextValidNonce, "Old transaction");
45 }
46
```

Proof of Concept:

1. Confuse wormhole chain ID with network chain ID
2. Initialize the contract with a wrong wormhole chain ID
3. Execute transactions on the protocol
4. Validate wormhole message emitter does not work as intended

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

Creating a Chain ID whitelist with all the possible Chain IDs or having it hardcoded within the contract is recommended.

Remediation Plan:

SOLVED: The Affine DeFi team solved the issue in commit: 06d6bc37fa80f0fdf794a8cb93e8100288d065e0

3.3 (HAL-03) POSSIBLE UNPREDICTABILITY BETWEEN L2 AND L1 RATIOS - LOW

Description:

When setLayerRatios function is used to update the ratio between L1 and L2, an invalid total ratio can be set (more than 100%). Hence, the rebalancer bot could not properly work in those cases.

Code Location:

Listing 6: L2Vault.sol (Lines 450,451)

```
function setLayerRatios(uint256 _l1Ratio, uint256 _l2Ratio)
↳ external onlyGovernance {
    l1Ratio = _l1Ratio;
    l2Ratio = _l2Ratio;
}
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

When setting the ratios, making sure the total ratio is equal to 100% is recommended.

Remediation Plan:

RISK ACCEPTED: The Affine DeFi team accepted the risk of this finding.

3.4 (HAL-04) FUNCTION DOES NOT CHECK THE TOKEN BALANCE BEFORE AND AFTER A CALL - LOW

Description:

Whenever the exit function is used, the contract should check the token balance before and after the call. So, the exact amount of tokens sent can be properly checked.

Code Location:

Listing 7: BridgeEscrow.sol (Line 64)

```
60 function l1ClearFund(uint256 amount, bytes calldata exitProof)
↳ external {
61     require(msg.sender == wormholeRouter, "Only wormhole router");
62
63     // Exit tokens, after that the withdrawn tokens from L2 will
↳ be reflected in L1 BridgeEscrow.
64     rootChainManager.exit(exitProof);
65
66     // Transfer exited tokens to L1 Vault.
67     uint256 balance = token.balanceOf(address(this));
68     require(balance >= amount, "Funds not received");
69
70     IL1Vault l1Vault = IL1Vault(vault);
71     token.safeTransfer(address(l1Vault), balance);
72
73     l1Vault.afterReceive();
74 }
75
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Checking the token balance before and after the exit call is recommended.

Remediation Plan:

SOLVED: The Affine DeFi team solved the issue in commit:
06d6bc37fa80f0fdf794a8cb93e8100288d065e0

3.5 (HAL-05) LACK OF PROPER SLIPPAGE PROTECTION - INFORMATIONAL }

Description:

Within the _claimAndSellRewards function, the slippage protection of the transaction is set to zero. Hence, if there is tiny liquidity, there is a high risk of losing part of the investment.

Code Location:

Listing 8: L1CompoundStrategy.sol (Line 127)

```
function _claimAndSellRewards() internal {
    comptroller.claimComp(address(this));
    if (rewardToken != address(cToken)) {
        uint256 rewardTokenBalance = balanceOfRewardToken();
        if (rewardTokenBalance >= minRewardToSell) {
            _sellRewardTokenForWant(rewardTokenBalance, 0);
        }
    }
    return;
}

```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Setting at least 5% slippage protection is recommended.

Remediation Plan:

SOLVED: The Affine DeFi team solved the issue in commit:
06d6bc37fa80f0fdf794a8cb93e8100288d065e0

3.6 (HAL-06) POSSIBLE MISUSE OF CHAIN ID - INFORMATIONAL

Description:

When initializing the wormhole router, the wormhole chain ID can be misused. As can be wrongly set due to confusion with the different deployed chain IDs.

As mentioned on HAL02, you can check the Wormhole Chain IDs on each chain, which is not the same as the network chain ID and can be easily confused.

Code Location:

Listing 9: L2WormholeRouter.sol (Line 26)

```
function initialize(IWormhole _wormhole, L2Vault _vault, address
↳ _otherLayerRouter, uint16 _otherLayerChainId)
    external
    initializer
{
    wormhole = _wormhole;
    vault = _vault;
    governance = vault.governance();
    otherLayerRouter = _otherLayerRouter;
    otherLayerChainId = _otherLayerChainId;
}

```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

As mentioned on HAL02, creating a Chain ID whitelist with all the possible Chain IDs or having it hardcoded within the contract is recommended.

Remediation Plan:

SOLVED: The Affine DeFi team solved the issue in commit: 06d6bc37fa80f0fdf794a8cb93e8100288d065e0

AUTOMATED TESTING

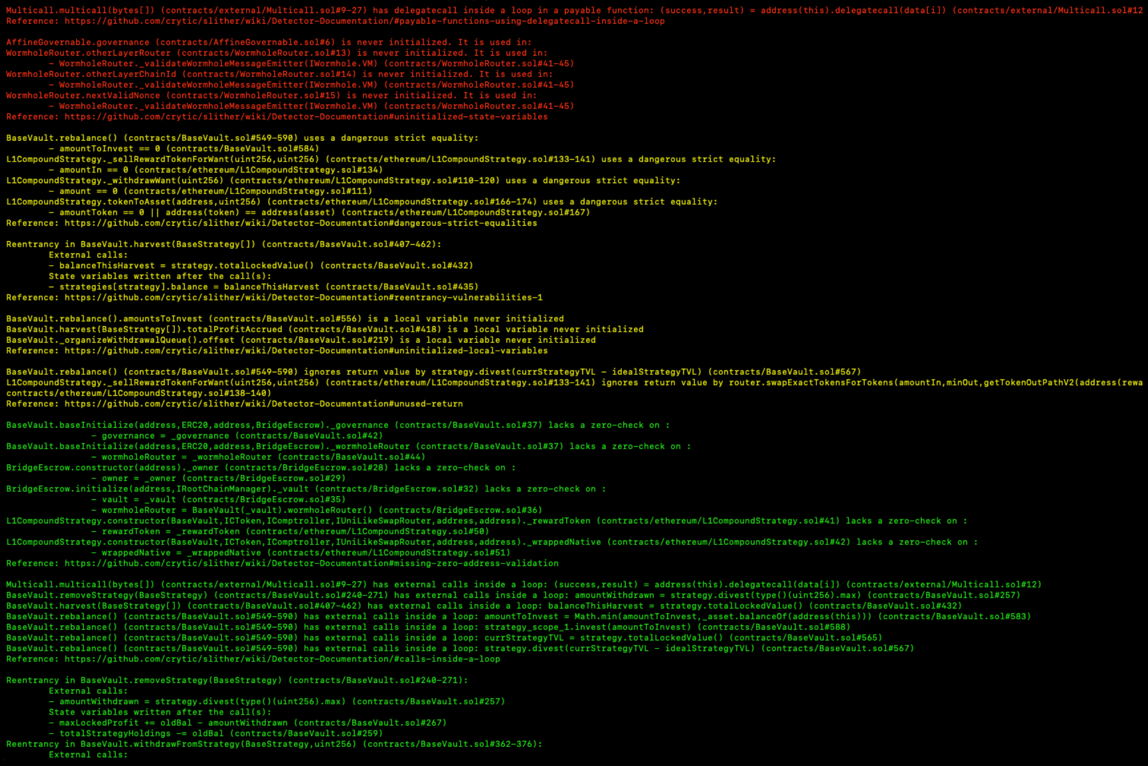
4.1 STATIC ANALYSIS REPORT

Description:

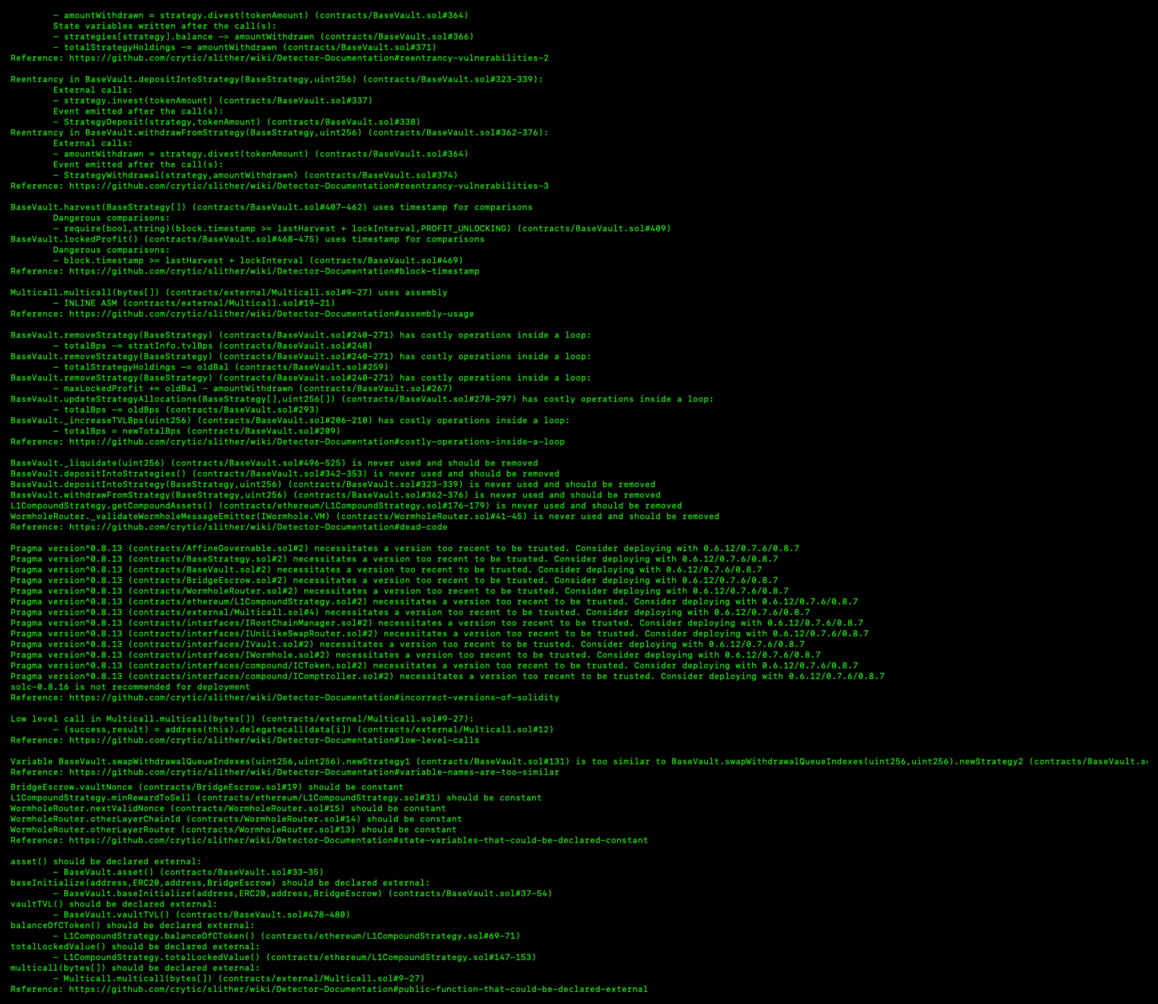
Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

src/ethereum/L1CompoundStrategy.sol



```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
L1CompoundStrategy._sellRewardTokenForWant(uint256,uint256) (contracts/ethereum/L1CompoundStrategy.sol#133-141) uses a dangerous strict equality:
        - amountIn == 0 (contracts/ethereum/L1CompoundStrategy.sol#134)
L1CompoundStrategy._withdrawWant(uint256) (contracts/ethereum/L1CompoundStrategy.sol#110-120) uses a dangerous strict equality:
        - amount == 0 (contracts/ethereum/L1CompoundStrategy.sol#111)
L1CompoundStrategy.tokenToAsset(address,uint256) (contracts/ethereum/L1CompoundStrategy.sol#166-174) uses a dangerous strict equality:
        - amountToken == 0 || address(token) == address(asset) (contracts/ethereum/L1CompoundStrategy.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L1CompoundStrategy._sellRewardTokenForWant(uint256,uint256) (contracts/ethereum/L1CompoundStrategy.sol#133-141) ignores return value by router.swapExactTokensForTokens(amountIn,minOut,getTokenOutPathV2(address(rewa
contracts/ethereum/L1CompoundStrategy.sol#138-140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
L1CompoundStrategy.constructor(BaseVault,ICToken,IComptroller,IUniLikeSwapRouter,address,address)._rewardToken (contracts/ethereum/L1CompoundStrategy.sol#41) lacks a zero-check on :
                - rewardToken = _rewardToken (contracts/ethereum/L1CompoundStrategy.sol#50)
L1CompoundStrategy.constructor(BaseVault,ICToken,IComptroller,IUniLikeSwapRouter,address,address)._wrappedNative (contracts/ethereum/L1CompoundStrategy.sol#42) lacks a zero-check on :
                - wrappedNative = _wrappedNative (contracts/ethereum/L1CompoundStrategy.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
```



```
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
L1CompoundStrategy.getCompoundAssets() (contracts/ethereum/L1CompoundStrategy.sol#176-179) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/ethereum/L1CompoundStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IUniLikeSwapRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/compound/ICToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/compound/IComptroller.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.s
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
L1CompoundStrategy.minRewardToSell (contracts/ethereum/L1CompoundStrategy.sol#31) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
baseInitialize(address,ERC20,address,BridgeEscrow) should be declared external:
        - BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow) (contracts/BaseVault.sol#37-54)
vaultTVL() should be declared external:
        - BaseVault.vaultTVL() (contracts/BaseVault.sol#478-480)
balanceOfCToken() should be declared external:
        - L1CompoundStrategy.balanceOfCToken() (contracts/ethereum/L1CompoundStrategy.sol#69-71)
totalLockedValue() should be declared external:
        - L1CompoundStrategy.totalLockedValue() (contracts/ethereum/L1CompoundStrategy.sol#147-153)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/ethereum/L1Vault.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reentrancy in L1Vault.sendTVL() (contracts/ethereum/L1Vault.sol#63-75):
        External calls:
        - L1WormholeRouter(wormholeRouter).reportTVL(tvl,received) (contracts/ethereum/L1Vault.sol#67)
        State variables written after the call(s):
        - received = false (contracts/ethereum/L1Vault.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L1WormholeRouter.reportTVL(uint256,bool) (contracts/ethereum/L1WormholeRouter.sol#29-38) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/ethereum/L1WormholeRout
L1WormholeRouter.reportTransferredFund(uint256) (contracts/ethereum/L1WormholeRouter.sol#40-46) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/ethereum/L1Wormh
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
L1Vault.initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address)._predicate (contracts/ethereum/L1Vault.sol#37) lacks a zero-check on :
                - predicate = _predicate (contracts/ethereum/L1Vault.sol#43)
L1WormholeRouter.initialize(IWormhole,L1Vault,address,uint16)._otherLayerRouter (contracts/ethereum/L1WormholeRouter.sol#18) lacks a zero-check on :
                - otherLayerRouter = _otherLayerRouter (contracts/ethereum/L1WormholeRouter.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in L1Vault._transferFundsToL2(uint256) (contracts/ethereum/L1Vault.sol#88-95):
        External calls:
        - chainManager.depositFor(address(bridgeEscrow),address(_asset),abi.encodePacked(amount)) (contracts/ethereum/L1Vault.sol#90)
        - L1WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/ethereum/L1Vault.sol#93)
        Event emitted after the call(s):
        - FundTransferToL2(amount) (contracts/ethereum/L1Vault.sol#94)
Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in L1Vault.processFundRequest(uint256) (contracts/ethereum/L1Vault.sol#78-83):
        External calls:
        - _liquidate(amountRequested) (contracts/ethereum/L1Vault.sol#80)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferFundsToL2(amountToSend) (contracts/ethereum/L1Vault.sol#82)
                - chainManager.depositFor(address(bridgeEscrow),address(_asset),abi.encodePacked(amount)) (contracts/ethereum/L1Vault.sol#90)
                - L1WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/ethereum/L1Vault.sol#93)
        Event emitted after the call(s):
        - FundTransferToL2(amount) (contracts/ethereum/L1Vault.sol#94)
                - _transferFundsToL2(amountToSend) (contracts/ethereum/L1Vault.sol#82)
Reentrancy in L1WormholeRouter.receiveFunds(bytes,bytes) (contracts/ethereum/L1WormholeRouter.sol#50-62):
        External calls:
        - vault.bridgeEscrow().l1ClearFund(amount,data) (contracts/ethereum/L1WormholeRouter.sol#60)
        Event emitted after the call(s):
        - TransferFromL2(amount) (contracts/ethereum/L1WormholeRouter.sol#61)
Reentrancy in L1Vault.sendTVL() (contracts/ethereum/L1Vault.sol#63-75):
        External calls:
        - L1WormholeRouter(wormholeRouter).reportTVL(tvl,received) (contracts/ethereum/L1Vault.sol#67)
        Event emitted after the call(s):
        - SendTVL(tvl) (contracts/ethereum/L1Vault.sol#74)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

L1Vault._msgData() (contracts/ethereum/L1Vault.sol#50-52) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/ethereum/L1Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.16 (contracts/ethereum/L1WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

L1Vault (contracts/ethereum/L1Vault.sol#18-103) should inherit from IL1Vault (contracts/interfaces/IVault.sol#4-6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address) should be declared external:
        - L1Vault.initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address) (contracts/ethereum/L1Vault.sol#31-44)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/ethereum/L1WormholeRouter.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reentrancy in L1Vault.sendTVL() (contracts/ethereum/L1Vault.sol#63-75):
        External calls:
        - L1WormholeRouter(wormholeRouter).reportTVL(tvl,received) (contracts/ethereum/L1Vault.sol#67)
        State variables written after the call(s):
        - received = false (contracts/ethereum/L1Vault.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L1WormholeRouter.reportTVL(uint256,bool) (contracts/ethereum/L1WormholeRouter.sol#29-38) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/ethereum/L1WormholeRout
L1WormholeRouter.reportTransferredFund(uint256) (contracts/ethereum/L1WormholeRouter.sol#40-46) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/ethereum/L1Wormh
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
L1Vault.initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address)._predicate (contracts/ethereum/L1Vault.sol#37) lacks a zero-check on :
                - predicate = _predicate (contracts/ethereum/L1Vault.sol#43)
L1WormholeRouter.initialize(IWormhole,L1Vault,address,uint16)._otherLayerRouter (contracts/ethereum/L1WormholeRouter.sol#18) lacks a zero-check on :
                - otherLayerRouter = _otherLayerRouter (contracts/ethereum/L1WormholeRouter.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in L1Vault._transferFundsToL2(uint256) (contracts/ethereum/L1Vault.sol#88-95):
        External calls:
        - chainManager.depositFor(address(bridgeEscrow),address(_asset),abi.encodePacked(amount)) (contracts/ethereum/L1Vault.sol#90)
        - L1WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/ethereum/L1Vault.sol#93)
        Event emitted after the call(s):
        - FundTransferToL2(amount) (contracts/ethereum/L1Vault.sol#94)
Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in L1Vault.processFundRequest(uint256) (contracts/ethereum/L1Vault.sol#78-83):
        External calls:
        - _liquidate(amountRequested) (contracts/ethereum/L1Vault.sol#80)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferFundsToL2(amountToSend) (contracts/ethereum/L1Vault.sol#82)
                - chainManager.depositFor(address(bridgeEscrow),address(_asset),abi.encodePacked(amount)) (contracts/ethereum/L1Vault.sol#90)
                - L1WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/ethereum/L1Vault.sol#93)
        Event emitted after the call(s):
        - FundTransferToL2(amount) (contracts/ethereum/L1Vault.sol#94)
                - _transferFundsToL2(amountToSend) (contracts/ethereum/L1Vault.sol#82)
Reentrancy in L1WormholeRouter.receiveFunds(bytes,bytes) (contracts/ethereum/L1WormholeRouter.sol#50-62):
        External calls:
        - vault.bridgeEscrow().l1ClearFund(amount,data) (contracts/ethereum/L1WormholeRouter.sol#60)
        Event emitted after the call(s):
        - TransferFromL2(amount) (contracts/ethereum/L1WormholeRouter.sol#61)
Reentrancy in L1Vault.sendTVL() (contracts/ethereum/L1Vault.sol#63-75):
        External calls:
        - L1WormholeRouter(wormholeRouter).reportTVL(tvl,received) (contracts/ethereum/L1Vault.sol#67)
        Event emitted after the call(s):
        - SendTVL(tvl) (contracts/ethereum/L1Vault.sol#74)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

L1Vault._msgData() (contracts/ethereum/L1Vault.sol#50-52) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/ethereum/L1Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.16 (contracts/ethereum/L1WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

L1Vault (contracts/ethereum/L1Vault.sol#18-103) should inherit from IL1Vault (contracts/interfaces/IVault.sol#4-6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address) should be declared external:
        - L1Vault.initialize(address,ERC20,address,BridgeEscrow,IRootChainManager,address) (contracts/ethereum/L1Vault.sol#31-44)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

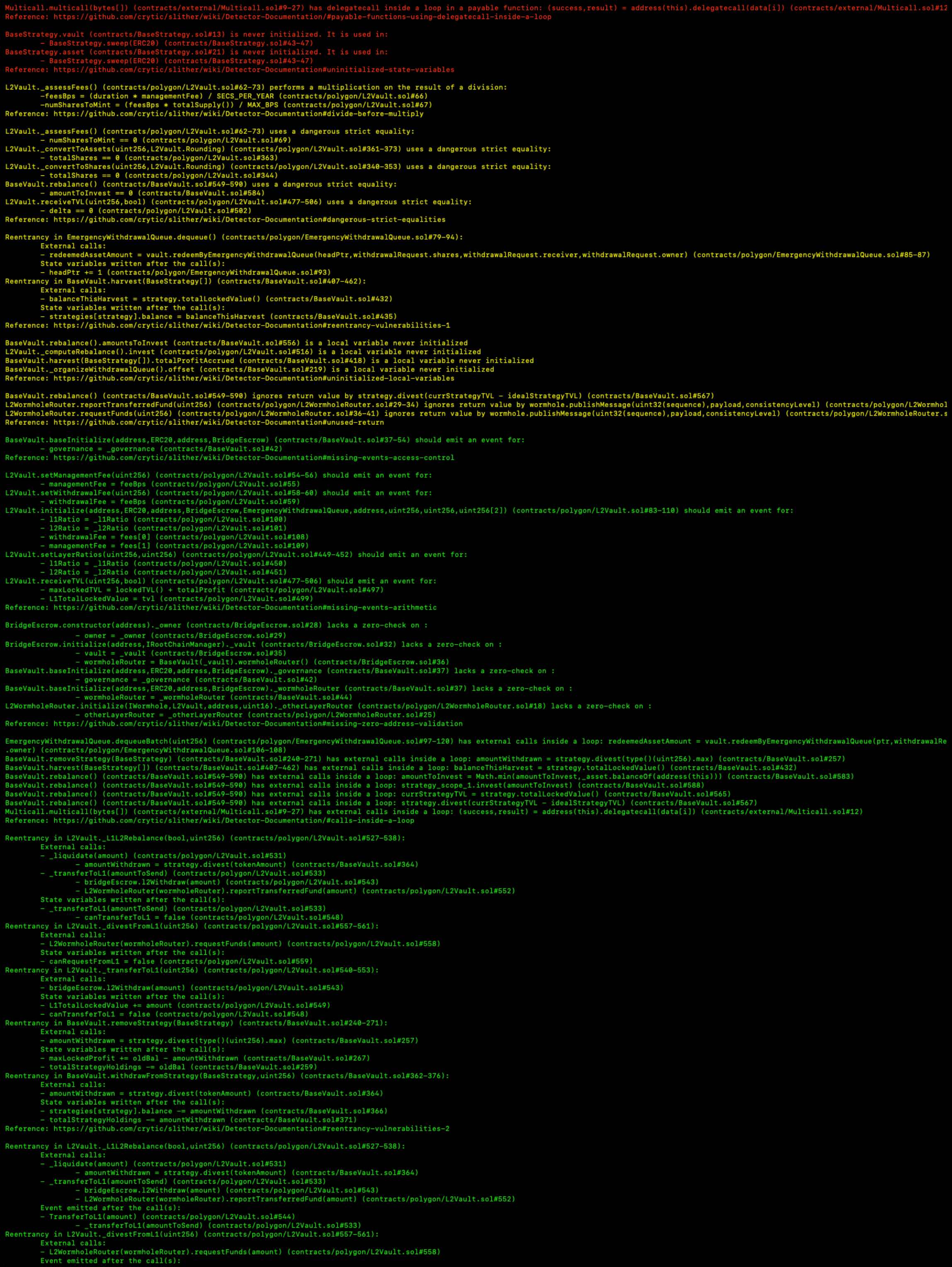
src/external/Multicall.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/Detailed.sol

```
Pragma version^0.8.13 (contracts/polygon/Detailed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

src/polygon/EmergencyWithdrawalQueue.sol



```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) performs a multiplication on the result of a division:
        -feesBps = (duration * managementFee) / SECS_PER_YEAR (contracts/polygon/L2Vault.sol#66)
        -numSharesToMint = (feesBps * totalSupply()) / MAX_BPS (contracts/polygon/L2Vault.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses a dangerous strict equality:
        - numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#344)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) uses a dangerous strict equality:
        - delta == 0 (contracts/polygon/L2Vault.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
        State variables written after the call(s):
        - headPtr += 1 (contracts/polygon/EmergencyWithdrawalQueue.sol#93)
Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
L2Vault._computeRebalance().invest (contracts/polygon/L2Vault.sol#516) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L2WormholeRouter.reportTransferredFund(uint256) (contracts/polygon/L2WormholeRouter.sol#29-34) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2Wormhol
L2WormholeRouter.requestFunds(uint256) (contracts/polygon/L2WormholeRouter.sol#36-41) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2WormholeRouter.s
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow) (contracts/BaseVault.sol#37-54) should emit an event for:
        - governance = _governance (contracts/BaseVault.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

L2Vault.setManagementFee(uint256) (contracts/polygon/L2Vault.sol#54-56) should emit an event for:
        - managementFee = feeBps (contracts/polygon/L2Vault.sol#55)
L2Vault.setWithdrawalFee(uint256) (contracts/polygon/L2Vault.sol#58-60) should emit an event for:
        - withdrawalFee = feeBps (contracts/polygon/L2Vault.sol#59)
L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#100)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#101)
        - withdrawalFee = fees[0] (contracts/polygon/L2Vault.sol#108)
        - managementFee = fees[1] (contracts/polygon/L2Vault.sol#109)
L2Vault.setLayerRatios(uint256,uint256) (contracts/polygon/L2Vault.sol#449-452) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#450)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#451)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) should emit an event for:
        - maxLockedTVL = lockedTVL() + totalProfit (contracts/polygon/L2Vault.sol#497)
        - L1TotalLockedValue = tvl (contracts/polygon/L2Vault.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
L2WormholeRouter.initialize(IWormhole,L2Vault,address,uint16)._otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#18) lacks a zero-check on :
                - otherLayerRouter = _otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has external calls inside a loop: redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalRe
.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        State variables written after the call(s):
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        State variables written after the call(s):
        - canRequestFromL1 = false (contracts/polygon/L2Vault.sol#559)
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
        External calls:
        - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
        State variables written after the call(s):
        - L1TotalLockedValue += amount (contracts/polygon/L2Vault.sol#549)
        - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        Event emitted after the call(s):
        - TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
                - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        Event emitted after the call(s):
```

```
        - RequestFromL1(amount) (contracts/polygon/L2Vault.sol#560)
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
        External calls:
        - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
        Event emitted after the call(s):
        - TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
        Event emitted after the call(s):
        - EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#89-91)
Reentrancy in EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
        Event emitted after the call(s):
        - EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#110-112)
Reentrancy in L2WormholeRouter.receiveFunds(bytes) (contracts/polygon/L2WormholeRouter.sol#45-56):
        External calls:
        - vault.bridgeEscrow().l2ClearFund(amount) (contracts/polygon/L2WormholeRouter.sol#54)
        Event emitted after the call(s):
        - TransferFromL1(amount) (contracts/polygon/L2WormholeRouter.sol#55)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94) uses timestamp for comparisons
        Dangerous comparisons:
        - redeemedAssetAmount > 0 (contracts/polygon/EmergencyWithdrawalQueue.sol#88)
L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses timestamp for comparisons
        Dangerous comparisons:
        - numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault.redeemByEmergencyWithdrawalQueue(uint256,uint256,address,address) (contracts/polygon/L2Vault.sol#209-236) uses timestamp for comparisons
        Dangerous comparisons:
        - balanceOf(owner) < shares (contracts/polygon/L2Vault.sol#218)
L2Vault.redeem(uint256,address,address) (contracts/polygon/L2Vault.sol#239-274) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#240-243)
L2Vault.withdraw(uint256,address,address) (contracts/polygon/L2Vault.sol#277-317) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#283-286)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses timestamp for comparisons
        Dangerous comparisons:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#344)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses timestamp for comparisons
        Dangerous comparisons:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault.lockedTVL() (contracts/polygon/L2Vault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastTVLUpdate + lockInterval (contracts/polygon/L2Vault.sol#469)
L2Vault.detailedPrice() (contracts/polygon/L2Vault.sol#580-584) uses timestamp for comparisons
        Dangerous comparisons:
        - totalSupply() > 0 (contracts/polygon/L2Vault.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has costly operations inside a loop:
        - delete queue[ptr] (contracts/polygon/EmergencyWithdrawalQueue.sol#103)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._assessFees() (contracts/BaseVault.sol#531) is never used and should be removed
L2Vault._msgData() (contracts/polygon/L2Vault.sol#122-129) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/Detailed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/EmergencyWithdrawalQueue.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/L2Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/polygon/L2WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

L2Vault (contracts/polygon/L2Vault.sol#29-589) should inherit from IL2Vault (contracts/interfaces/IVault.sol#8-10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Redundant expression "receiver (contracts/polygon/L2Vault.sol#414)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Redundant expression "receiver (contracts/polygon/L2Vault.sol#420)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
        - L2Vault.asset() (contracts/polygon/L2Vault.sol#149-151)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
linkVault(L2Vault) should be declared external:
        - EmergencyWithdrawalQueue.linkVault(L2Vault) (contracts/polygon/EmergencyWithdrawalQueue.sol#48-57)
totalDebt() should be declared external:
        - EmergencyWithdrawalQueue.totalDebt() (contracts/polygon/EmergencyWithdrawalQueue.sol#65-67)
initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) should be declared external:
        - L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110)
convertToShares(uint256) should be declared external:
        - L2Vault.convertToShares(uint256) (contracts/polygon/L2Vault.sol#335-337)
convertToAssets(uint256) should be declared external:
        - L2Vault.convertToAssets(uint256) (contracts/polygon/L2Vault.sol#356-358)
previewRedeem(uint256) should be declared external:
        - L2Vault.previewRedeem(uint256) (contracts/polygon/L2Vault.sol#391-393)
maxDeposit(address) should be declared external:
        - L2Vault.maxDeposit(address) (contracts/polygon/L2Vault.sol#413-416)
maxMint(address) should be declared external:
        - L2Vault.maxMint(address) (contracts/polygon/L2Vault.sol#419-422)
maxRedeem(address) should be declared external:
        - L2Vault.maxRedeem(address) (contracts/polygon/L2Vault.sol#425-427)
maxWithdraw(address) should be declared external:
        - L2Vault.maxWithdraw(address) (contracts/polygon/L2Vault.sol#430-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/ERC4626Router.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/ERC4626Router.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/ERC4626RouterBase.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
depositMax(IERC4626,address,uint256) should be declared external:
        - ERC4626Router.depositMax(IERC4626,address,uint256) (contracts/polygon/ERC4626Router.sol#45-52)
redeemMax(IERC4626,address,uint256) should be declared external:
        - ERC4626Router.redeemMax(IERC4626,address,uint256) (contracts/polygon/ERC4626Router.sol#54-59)
approve(ERC20,address,uint256) should be declared external:
        - ERC4626Router.approve(ERC20,address,uint256) (contracts/polygon/ERC4626Router.sol#61-63)
mint(IERC4626,address,uint256,uint256) should be declared external:
        - ERC4626RouterBase.mint(IERC4626,address,uint256,uint256) (contracts/polygon/ERC4626RouterBase.sol#24-33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/ERC4626RouterBase.sol

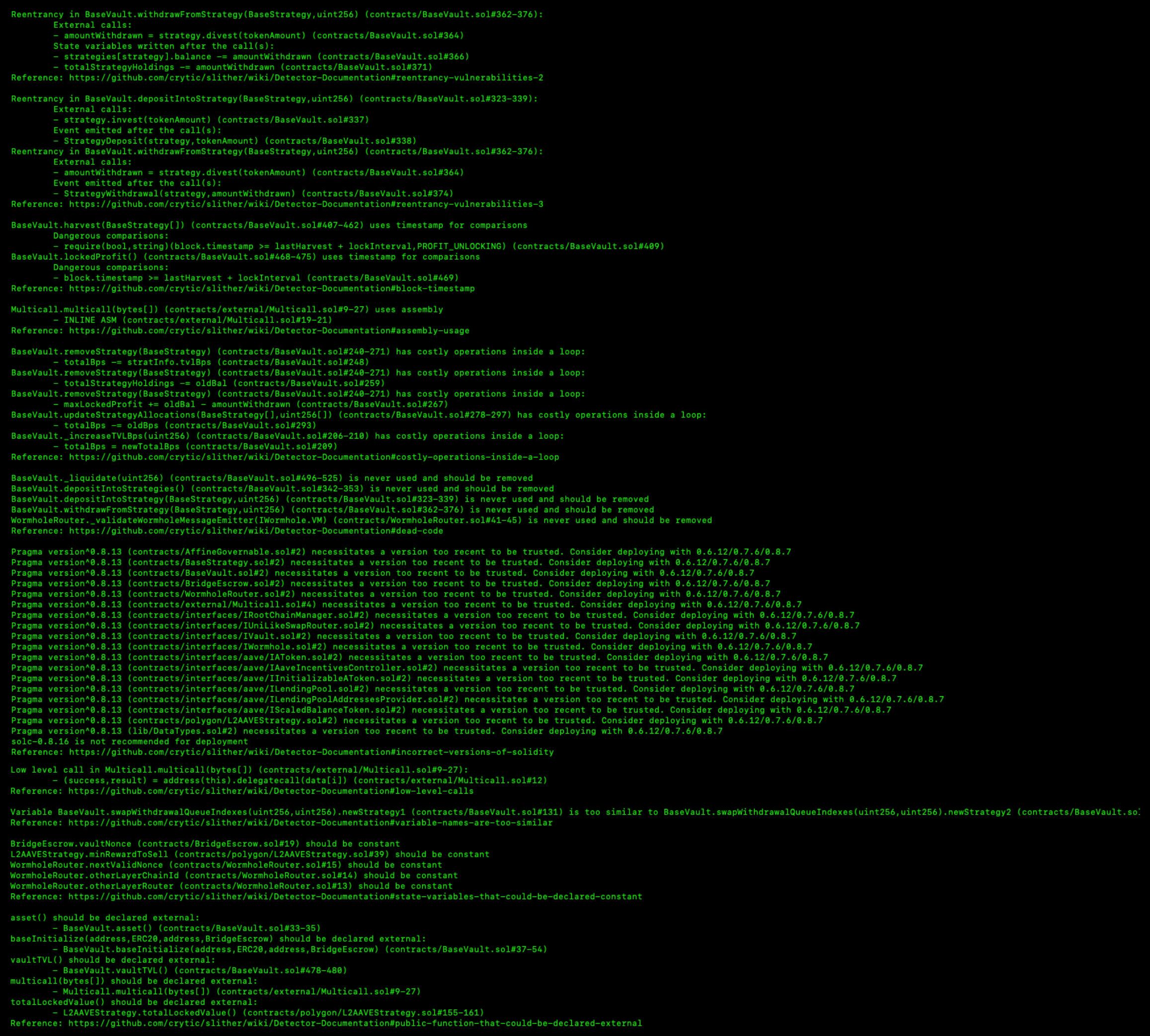
```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/ERC4626RouterBase.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
mint(IERC4626,address,uint256,uint256) should be declared external:
        - ERC4626RouterBase.mint(IERC4626,address,uint256,uint256) (contracts/polygon/ERC4626RouterBase.sol#24-33)
deposit(IERC4626,address,uint256,uint256) should be declared external:
        - ERC4626RouterBase.deposit(IERC4626,address,uint256,uint256) (contracts/polygon/ERC4626RouterBase.sol#35-44)
withdraw(IERC4626,address,uint256,uint256) should be declared external:
        - ERC4626RouterBase.withdraw(IERC4626,address,uint256,uint256) (contracts/polygon/ERC4626RouterBase.sol#46-55)
redeem(IERC4626,address,uint256,uint256) should be declared external:
        - ERC4626RouterBase.redeem(IERC4626,address,uint256,uint256) (contracts/polygon/ERC4626RouterBase.sol#57-66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/Forwarder.sol

```
Pragma version^0.8.13 (contracts/polygon/Forwarder.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

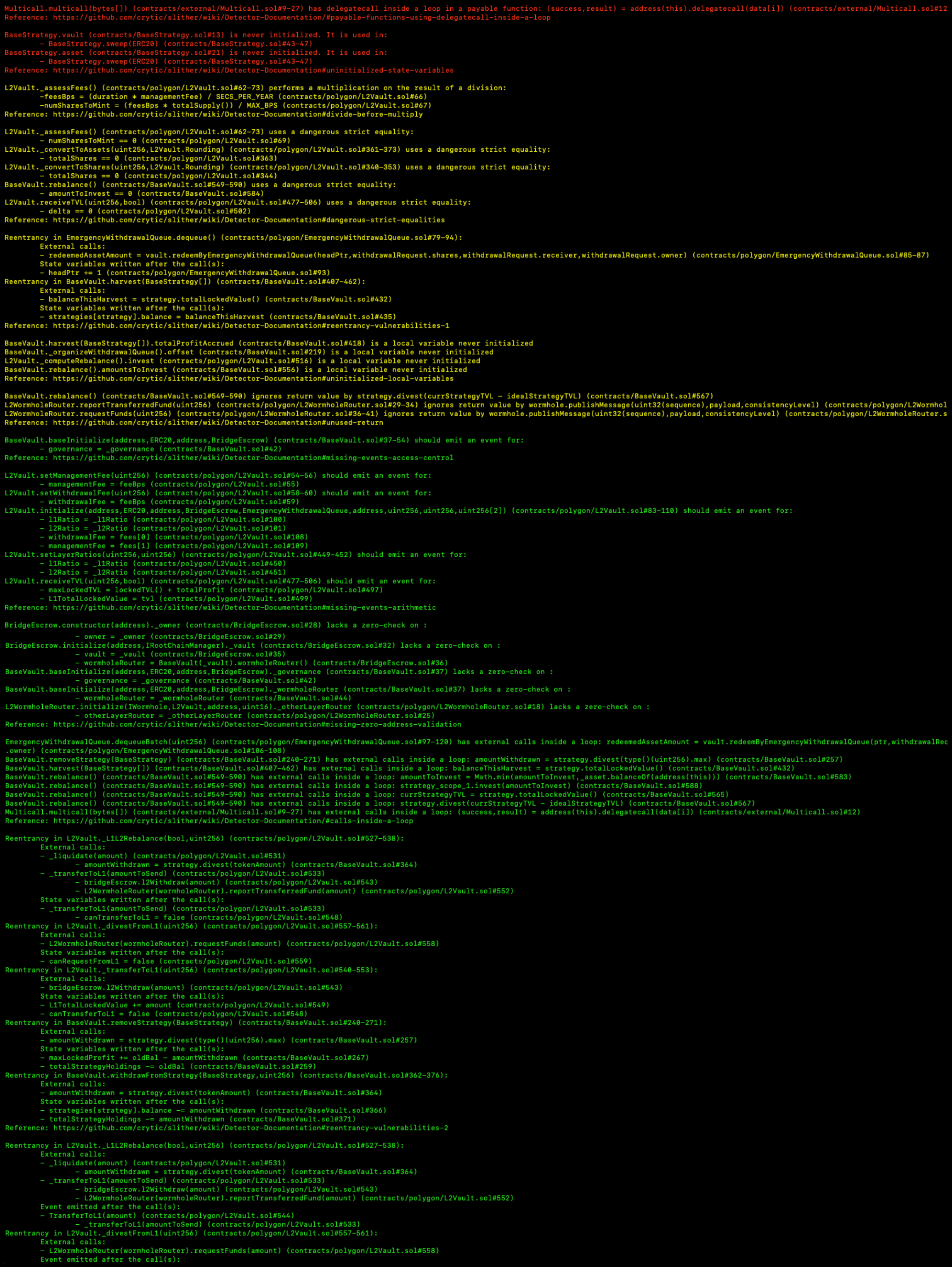
src/polygon/L2AAVEStrategy.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
L2AAVEStrategy._sellRewardTokenForWant(uint256,uint256) (contracts/polygon/L2AAVEStrategy.sol#141-149) uses a dangerous strict equality:
        - amountIn == 0 (contracts/polygon/L2AAVEStrategy.sol#142)
L2AAVEStrategy._withdrawWant(uint256) (contracts/polygon/L2AAVEStrategy.sol#122-128) uses a dangerous strict equality:
        - amount == 0 (contracts/polygon/L2AAVEStrategy.sol#123)
L2AAVEStrategy.tokenToAsset(address,uint256) (contracts/polygon/L2AAVEStrategy.sol#175-183) uses a dangerous strict equality:
        - amountToken == 0 || address(token) == address(asset) (contracts/polygon/L2AAVEStrategy.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L2AAVEStrategy._withdrawWant(uint256) (contracts/polygon/L2AAVEStrategy.sol#122-128) ignores return value by lendingPool.withdraw(address(asset),amount,address(this)) (contracts/polygon/L2AAVEStrategy.sol#126)
L2AAVEStrategy._claimAndSellRewards() (contracts/polygon/L2AAVEStrategy.sol#130-139) ignores return value by incentivesController.claimRewards(getAaveAssets(),type()(uint256).max,address(this)) (contracts/polygon/L
L2AAVEStrategy._sellRewardTokenForWant(uint256,uint256) (contracts/polygon/L2AAVEStrategy.sol#141-149) ignores return value by router.swapExactTokensForTokens(amountIn,minOut,getTokenOutPathV2(address(rewardToken),
/polygon/L2AAVEStrategy.sol#146-148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
L2AAVEStrategy.constructor(BaseVault,address,address,address,address,address)._rewardToken (contracts/polygon/L2AAVEStrategy.sol#48) lacks a zero-check on :
                - rewardToken = _rewardToken (contracts/polygon/L2AAVEStrategy.sol#63)
L2AAVEStrategy.constructor(BaseVault,address,address,address,address,address)._wrappedNative (contracts/polygon/L2AAVEStrategy.sol#49) lacks a zero-check on :
                - wrappedNative = _wrappedNative (contracts/polygon/L2AAVEStrategy.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
```

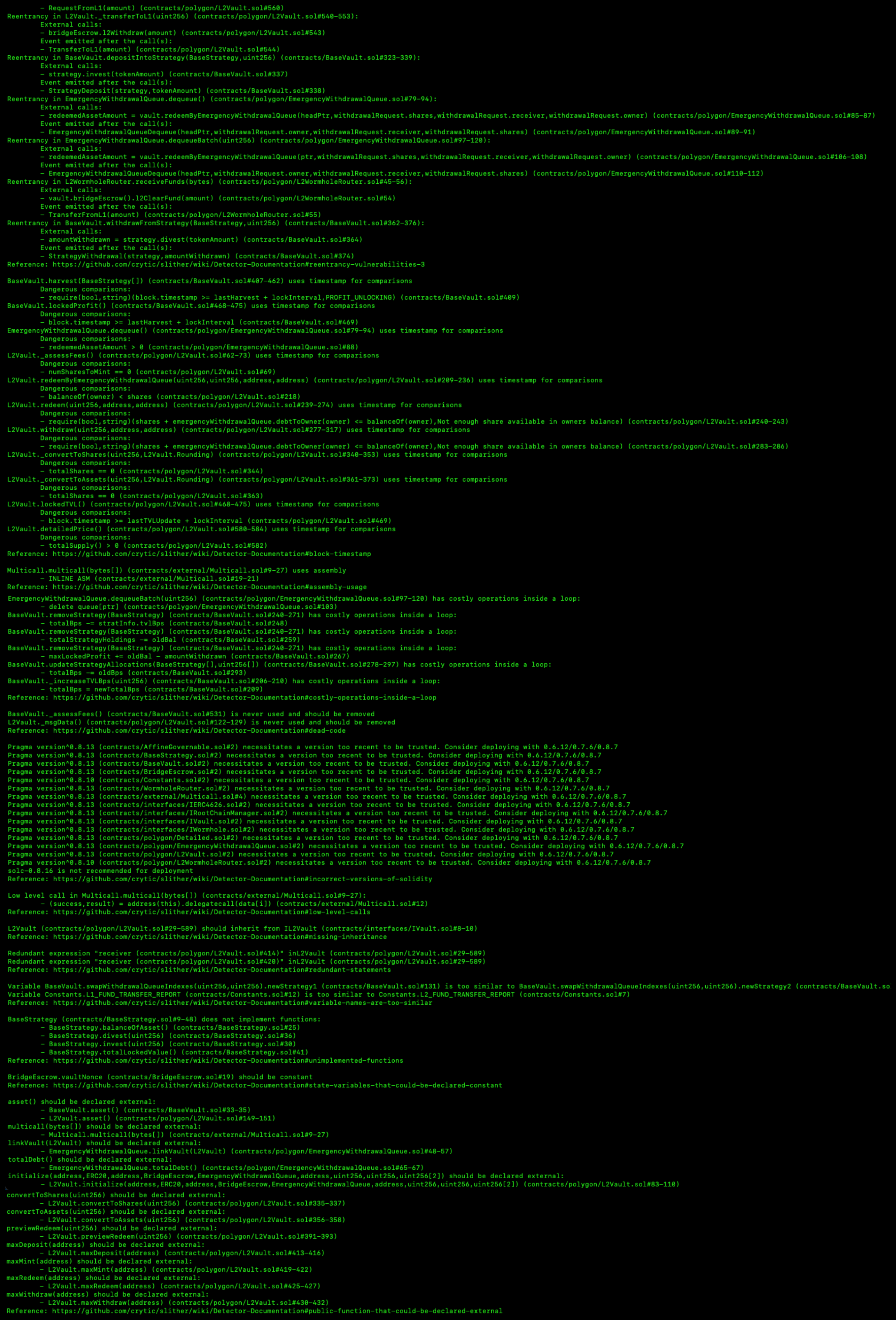


```
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IUniLikeSwapRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/IAToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/IAaveIncentivesController.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/IInitializableAToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/ILendingPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/ILendingPoolAddressesProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/aave/IScaledBalanceToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/L2AAVEStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (lib/DataTypes.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
L2AAVEStrategy.minRewardToSell (contracts/polygon/L2AAVEStrategy.sol#39) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
baseInitialize(address,ERC20,address,BridgeEscrow) should be declared external:
        - BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow) (contracts/BaseVault.sol#37-54)
vaultTVL() should be declared external:
        - BaseVault.vaultTVL() (contracts/BaseVault.sol#478-480)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
totalLockedValue() should be declared external:
        - L2AAVEStrategy.totalLockedValue() (contracts/polygon/L2AAVEStrategy.sol#155-161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/L2Vault.sol

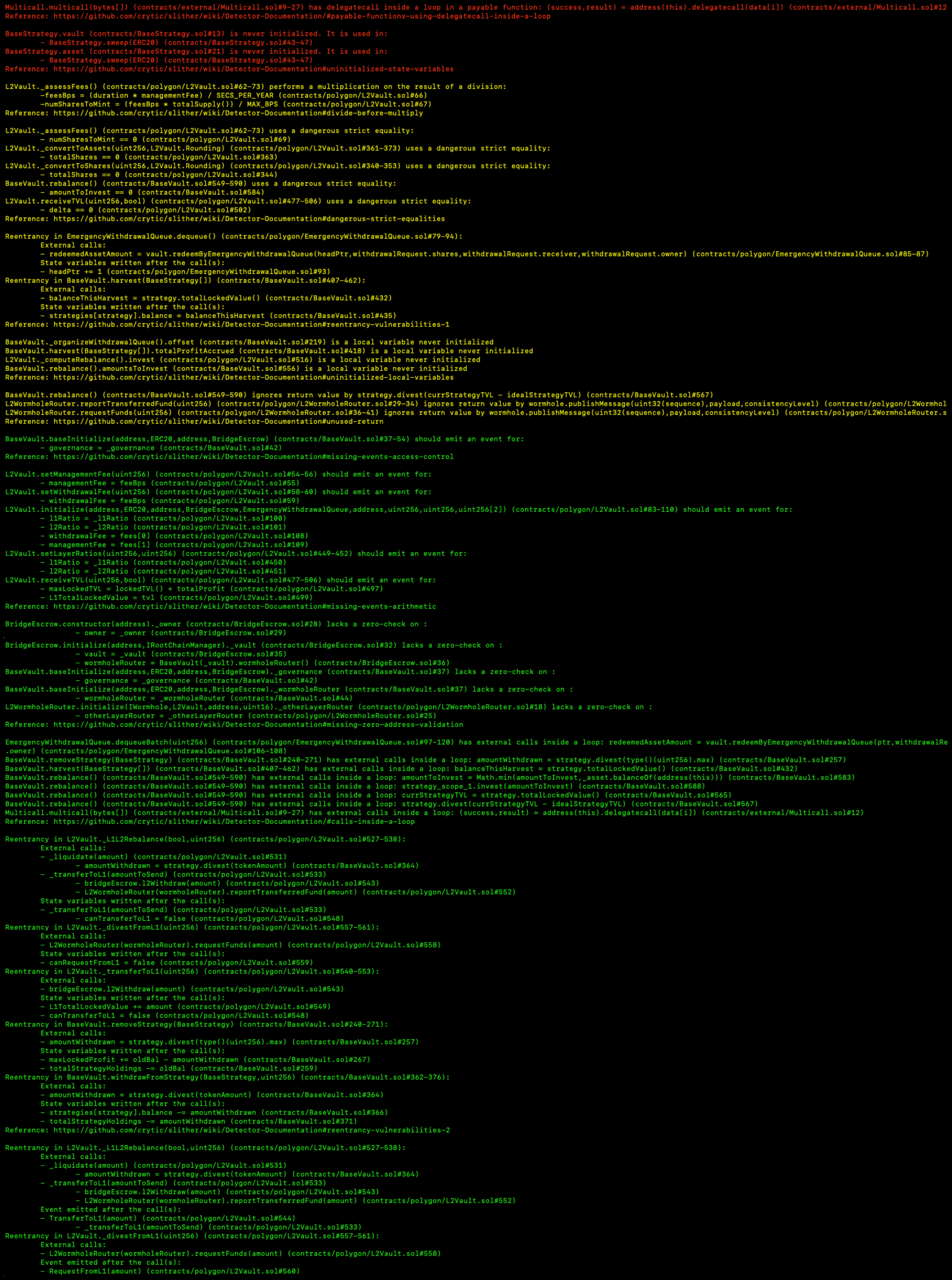


```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) performs a multiplication on the result of a division:
        -feesBps = (duration * managementFee) / SECS_PER_YEAR (contracts/polygon/L2Vault.sol#66)
        -numSharesToMint = (feesBps * totalSupply()) / MAX_BPS (contracts/polygon/L2Vault.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses a dangerous strict equality:
        - numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#344)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) uses a dangerous strict equality:
        - delta == 0 (contracts/polygon/L2Vault.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
        State variables written after the call(s):
        - headPtr += 1 (contracts/polygon/EmergencyWithdrawalQueue.sol#93)
Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
L2Vault._computeRebalance().invest (contracts/polygon/L2Vault.sol#516) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L2WormholeRouter.reportTransferredFund(uint256) (contracts/polygon/L2WormholeRouter.sol#29-34) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2Wormhol
L2WormholeRouter.requestFunds(uint256) (contracts/polygon/L2WormholeRouter.sol#36-41) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2WormholeRouter.s
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow) (contracts/BaseVault.sol#37-54) should emit an event for:
        - governance = _governance (contracts/BaseVault.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

L2Vault.setManagementFee(uint256) (contracts/polygon/L2Vault.sol#54-56) should emit an event for:
        - managementFee = feeBps (contracts/polygon/L2Vault.sol#55)
L2Vault.setWithdrawalFee(uint256) (contracts/polygon/L2Vault.sol#58-60) should emit an event for:
        - withdrawalFee = feeBps (contracts/polygon/L2Vault.sol#59)
L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#100)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#101)
        - withdrawalFee = fees[0] (contracts/polygon/L2Vault.sol#108)
        - managementFee = fees[1] (contracts/polygon/L2Vault.sol#109)
L2Vault.setLayerRatios(uint256,uint256) (contracts/polygon/L2Vault.sol#449-452) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#450)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#451)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) should emit an event for:
        - maxLockedTVL = lockedTVL() + totalProfit (contracts/polygon/L2Vault.sol#497)
        - L1TotalLockedValue = tvl (contracts/polygon/L2Vault.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
L2WormholeRouter.initialize(IWormhole,L2Vault,address,uint16)._otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#18) lacks a zero-check on :
                - otherLayerRouter = _otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has external calls inside a loop: redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalReq
.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        State variables written after the call(s):
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        State variables written after the call(s):
        - canRequestFromL1 = false (contracts/polygon/L2Vault.sol#559)
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
        External calls:
        - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
        State variables written after the call(s):
        - L1TotalLockedValue += amount (contracts/polygon/L2Vault.sol#549)
        - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        Event emitted after the call(s):
        - TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
                - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        Event emitted after the call(s):
```



```
        - RequestFromL1(amount) (contracts/polygon/L2Vault.sol#560)
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
        External calls:
        - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
        Event emitted after the call(s):
        - TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
        Event emitted after the call(s):
        - EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#89-91)
Reentrancy in EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
        Event emitted after the call(s):
        - EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#110-112)
Reentrancy in L2WormholeRouter.receiveFunds(bytes) (contracts/polygon/L2WormholeRouter.sol#45-56):
        External calls:
        - vault.bridgeEscrow().l2ClearFund(amount) (contracts/polygon/L2WormholeRouter.sol#54)
        Event emitted after the call(s):
        - TransferFromL1(amount) (contracts/polygon/L2WormholeRouter.sol#55)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94) uses timestamp for comparisons
        Dangerous comparisons:
        - redeemedAssetAmount > 0 (contracts/polygon/EmergencyWithdrawalQueue.sol#88)
L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses timestamp for comparisons
        Dangerous comparisons:
        - numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault.redeemByEmergencyWithdrawalQueue(uint256,uint256,address,address) (contracts/polygon/L2Vault.sol#209-236) uses timestamp for comparisons
        Dangerous comparisons:
        - balanceOf(owner) < shares (contracts/polygon/L2Vault.sol#218)
L2Vault.redeem(uint256,address,address) (contracts/polygon/L2Vault.sol#239-274) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#240-243)
L2Vault.withdraw(uint256,address,address) (contracts/polygon/L2Vault.sol#277-317) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#283-286)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses timestamp for comparisons
        Dangerous comparisons:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#344)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses timestamp for comparisons
        Dangerous comparisons:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault.lockedTVL() (contracts/polygon/L2Vault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastTVLUpdate + lockInterval (contracts/polygon/L2Vault.sol#469)
L2Vault.detailedPrice() (contracts/polygon/L2Vault.sol#580-584) uses timestamp for comparisons
        Dangerous comparisons:
        - totalSupply() > 0 (contracts/polygon/L2Vault.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has costly operations inside a loop:
        - delete queue[ptr] (contracts/polygon/EmergencyWithdrawalQueue.sol#103)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._assessFees() (contracts/BaseVault.sol#531) is never used and should be removed
L2Vault._msgData() (contracts/polygon/L2Vault.sol#122-129) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/Detailed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/EmergencyWithdrawalQueue.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/L2Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/polygon/L2WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

L2Vault (contracts/polygon/L2Vault.sol#29-589) should inherit from IL2Vault (contracts/interfaces/IVault.sol#8-10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Redundant expression "receiver (contracts/polygon/L2Vault.sol#414)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Redundant expression "receiver (contracts/polygon/L2Vault.sol#420)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
        - L2Vault.asset() (contracts/polygon/L2Vault.sol#149-151)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
linkVault(L2Vault) should be declared external:
        - EmergencyWithdrawalQueue.linkVault(L2Vault) (contracts/polygon/EmergencyWithdrawalQueue.sol#48-57)
totalDebt() should be declared external:
        - EmergencyWithdrawalQueue.totalDebt() (contracts/polygon/EmergencyWithdrawalQueue.sol#65-67)
initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) should be declared external:
        - L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110)
convertToShares(uint256) should be declared external:
        - L2Vault.convertToShares(uint256) (contracts/polygon/L2Vault.sol#335-337)
convertToAssets(uint256) should be declared external:
        - L2Vault.convertToAssets(uint256) (contracts/polygon/L2Vault.sol#356-358)
previewRedeem(uint256) should be declared external:
        - L2Vault.previewRedeem(uint256) (contracts/polygon/L2Vault.sol#391-393)
maxDeposit(address) should be declared external:
        - L2Vault.maxDeposit(address) (contracts/polygon/L2Vault.sol#413-416)
maxMint(address) should be declared external:
        - L2Vault.maxMint(address) (contracts/polygon/L2Vault.sol#419-422)
maxRedeem(address) should be declared external:
        - L2Vault.maxRedeem(address) (contracts/polygon/L2Vault.sol#425-427)
maxWithdraw(address) should be declared external:
        - L2Vault.maxWithdraw(address) (contracts/polygon/L2Vault.sol#430-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/polygon/L2WormholeRouter.sol



```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) performs a multiplication on the result of a division:
        -feesBps = (duration * managementFee) / SECS_PER_YEAR (contracts/polygon/L2Vault.sol#66)
        -numSharesToMint = (feesBps * totalSupply()) / MAX_BPS (contracts/polygon/L2Vault.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses a dangerous strict equality:
        - numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses a dangerous strict equality:
        - totalShares == 0 (contracts/polygon/L2Vault.sol#344)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) uses a dangerous strict equality:
        - delta == 0 (contracts/polygon/L2Vault.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
        External calls:
        - redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
        State variables written after the call(s):
        - headPtr += 1 (contracts/polygon/EmergencyWithdrawalQueue.sol#93)
Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
L2Vault._computeRebalance().invest (contracts/polygon/L2Vault.sol#516) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
L2WormholeRouter.reportTransferredFund(uint256) (contracts/polygon/L2WormholeRouter.sol#29-34) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2Wormhol
L2WormholeRouter.requestFunds(uint256) (contracts/polygon/L2WormholeRouter.sol#36-41) ignores return value by wormhole.publishMessage(uint32(sequence),payload,consistencyLevel) (contracts/polygon/L2WormholeRouter.s
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow) (contracts/BaseVault.sol#37-54) should emit an event for:
        - governance = _governance (contracts/BaseVault.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

L2Vault.setManagementFee(uint256) (contracts/polygon/L2Vault.sol#54-56) should emit an event for:
        - managementFee = feeBps (contracts/polygon/L2Vault.sol#55)
L2Vault.setWithdrawalFee(uint256) (contracts/polygon/L2Vault.sol#58-60) should emit an event for:
        - withdrawalFee = feeBps (contracts/polygon/L2Vault.sol#59)
L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#100)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#101)
        - withdrawalFee = fees[0] (contracts/polygon/L2Vault.sol#108)
        - managementFee = fees[1] (contracts/polygon/L2Vault.sol#109)
L2Vault.setLayerRatios(uint256,uint256) (contracts/polygon/L2Vault.sol#449-452) should emit an event for:
        - l1Ratio = _l1Ratio (contracts/polygon/L2Vault.sol#450)
        - l2Ratio = _l2Ratio (contracts/polygon/L2Vault.sol#451)
L2Vault.receiveTVL(uint256,bool) (contracts/polygon/L2Vault.sol#477-506) should emit an event for:
        - maxLockedTVL = lockedTVL() + totalProfit (contracts/polygon/L2Vault.sol#497)
        - L1TotalLockedValue = tvl (contracts/polygon/L2Vault.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
L2WormholeRouter.initialize(IWormhole,L2Vault,address,uint16)._otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#18) lacks a zero-check on :
                - otherLayerRouter = _otherLayerRouter (contracts/polygon/L2WormholeRouter.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has external calls inside a loop: redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalRe
.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        State variables written after the call(s):
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        State variables written after the call(s):
        - canRequestFromL1 = false (contracts/polygon/L2Vault.sol#559)
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
        External calls:
        - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
        State variables written after the call(s):
        - L1TotalLockedValue += amount (contracts/polygon/L2Vault.sol#549)
        - canTransferToL1 = false (contracts/polygon/L2Vault.sol#548)
Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in L2Vault._L1L2Rebalance(bool,uint256) (contracts/polygon/L2Vault.sol#527-538):
        External calls:
        - _liquidate(amount) (contracts/polygon/L2Vault.sol#531)
                - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
                - bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
                - L2WormholeRouter(wormholeRouter).reportTransferredFund(amount) (contracts/polygon/L2Vault.sol#552)
        Event emitted after the call(s):
        - TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
                - _transferToL1(amountToSend) (contracts/polygon/L2Vault.sol#533)
Reentrancy in L2Vault._divestFromL1(uint256) (contracts/polygon/L2Vault.sol#557-561):
        External calls:
        - L2WormholeRouter(wormholeRouter).requestFunds(amount) (contracts/polygon/L2Vault.sol#558)
        Event emitted after the call(s):
        - RequestFromL1(amount) (contracts/polygon/L2Vault.sol#560)
```

```
Reentrancy in L2Vault._transferToL1(uint256) (contracts/polygon/L2Vault.sol#540-553):
	External calls:
	- bridgeEscrow.l2Withdraw(amount) (contracts/polygon/L2Vault.sol#543)
	Event emitted after the call(s):
	- TransferToL1(amount) (contracts/polygon/L2Vault.sol#544)
Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
	External calls:
	- strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
	Event emitted after the call(s):
	- StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94):
	External calls:
	- redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(headPtr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#85-87)
	Event emitted after the call(s):
	- EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#89-91)
Reentrancy in EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120):
	External calls:
	- redeemedAssetAmount = vault.redeemByEmergencyWithdrawalQueue(ptr,withdrawalRequest.shares,withdrawalRequest.receiver,withdrawalRequest.owner) (contracts/polygon/EmergencyWithdrawalQueue.sol#106-108)
	Event emitted after the call(s):
	- EmergencyWithdrawalQueueDequeue(headPtr,withdrawalRequest.owner,withdrawalRequest.receiver,withdrawalRequest.shares) (contracts/polygon/EmergencyWithdrawalQueue.sol#110-112)
Reentrancy in L2WormholeRouter.receiveFunds(bytes) (contracts/polygon/L2WormholeRouter.sol#45-56):
	External calls:
	- vault.bridgeEscrow().l2ClearFund(amount) (contracts/polygon/L2WormholeRouter.sol#54)
	Event emitted after the call(s):
	- TransferFromL1(amount) (contracts/polygon/L2WormholeRouter.sol#55)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
	External calls:
	- amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
	Event emitted after the call(s):
	- StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
EmergencyWithdrawalQueue.dequeue() (contracts/polygon/EmergencyWithdrawalQueue.sol#79-94) uses timestamp for comparisons
	Dangerous comparisons:
	- redeemedAssetAmount > 0 (contracts/polygon/EmergencyWithdrawalQueue.sol#88)
L2Vault._assessFees() (contracts/polygon/L2Vault.sol#62-73) uses timestamp for comparisons
	Dangerous comparisons:
	- numSharesToMint == 0 (contracts/polygon/L2Vault.sol#69)
L2Vault.redeemByEmergencyWithdrawalQueue(uint256,uint256,address,address) (contracts/polygon/L2Vault.sol#209-236) uses timestamp for comparisons
	Dangerous comparisons:
	- balanceOf(owner) < shares (contracts/polygon/L2Vault.sol#218)
L2Vault.redeem(uint256,address,address) (contracts/polygon/L2Vault.sol#239-274) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#240-243)
L2Vault.withdraw(uint256,address,address) (contracts/polygon/L2Vault.sol#277-317) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(shares + emergencyWithdrawalQueue.debtToOwner(owner) <= balanceOf(owner),Not enough share available in owners balance) (contracts/polygon/L2Vault.sol#283-286)
L2Vault._convertToShares(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#340-353) uses timestamp for comparisons
	Dangerous comparisons:
	- totalShares == 0 (contracts/polygon/L2Vault.sol#344)
L2Vault._convertToAssets(uint256,L2Vault.Rounding) (contracts/polygon/L2Vault.sol#361-373) uses timestamp for comparisons
	Dangerous comparisons:
	- totalShares == 0 (contracts/polygon/L2Vault.sol#363)
L2Vault.lockedTVL() (contracts/polygon/L2Vault.sol#468-475) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp >= lastTVLUpdate + lockInterval (contracts/polygon/L2Vault.sol#469)
L2Vault.detailedPrice() (contracts/polygon/L2Vault.sol#580-584) uses timestamp for comparisons
	Dangerous comparisons:
	- totalSupply() > 0 (contracts/polygon/L2Vault.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
	- INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

EmergencyWithdrawalQueue.dequeueBatch(uint256) (contracts/polygon/EmergencyWithdrawalQueue.sol#97-120) has costly operations inside a loop:
	- delete queue[ptr] (contracts/polygon/EmergencyWithdrawalQueue.sol#103)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
	- totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
	- totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
	- maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
	- totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
	- totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._assessFees() (contracts/BaseVault.sol#531) is never used and should be removed
L2Vault._msgData() (contracts/polygon/L2Vault.sol#122-129) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IERC4626.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/Detailed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/EmergencyWithdrawalQueue.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/polygon/L2Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/polygon/L2WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
	- (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

L2Vault (contracts/polygon/L2Vault.sol#29-589) should inherit from IL2Vault (contracts/interfaces/IVault.sol#8-10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Redundant expression "receiver (contracts/polygon/L2Vault.sol#414)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Redundant expression "receiver (contracts/polygon/L2Vault.sol#420)" inL2Vault (contracts/polygon/L2Vault.sol#29-589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
	- BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
	- BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
	- BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
	- BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
	- BaseVault.asset() (contracts/BaseVault.sol#33-35)
	- L2Vault.asset() (contracts/polygon/L2Vault.sol#149-151)
multicall(bytes[]) should be declared external:
	- Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
linkVault(L2Vault) should be declared external:
	- EmergencyWithdrawalQueue.linkVault(L2Vault) (contracts/polygon/EmergencyWithdrawalQueue.sol#48-57)
totalDebt() should be declared external:
	- EmergencyWithdrawalQueue.totalDebt() (contracts/polygon/EmergencyWithdrawalQueue.sol#65-67)
initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) should be declared external:
	- L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110)
convertToShares(uint256) should be declared external:
```

```
BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

asset() should be declared external:
        - BaseVault.asset() (contracts/BaseVault.sol#33-35)
        - L2Vault.asset() (contracts/polygon/L2Vault.sol#149-151)
multicall(bytes[]) should be declared external:
        - Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27)
linkVault(L2Vault) should be declared external:
        - EmergencyWithdrawalQueue.linkVault(L2Vault) (contracts/polygon/EmergencyWithdrawalQueue.sol#48-57)
totalDebt() should be declared external:
        - EmergencyWithdrawalQueue.totalDebt() (contracts/polygon/EmergencyWithdrawalQueue.sol#65-67)
initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) should be declared external:
        - L2Vault.initialize(address,ERC20,address,BridgeEscrow,EmergencyWithdrawalQueue,address,uint256,uint256,uint256[2]) (contracts/polygon/L2Vault.sol#83-110)
convertToShares(uint256) should be declared external:
        - L2Vault.convertToShares(uint256) (contracts/polygon/L2Vault.sol#335-337)
convertToAssets(uint256) should be declared external:
        - L2Vault.convertToAssets(uint256) (contracts/polygon/L2Vault.sol#356-358)
previewRedeem(uint256) should be declared external:
        - L2Vault.previewRedeem(uint256) (contracts/polygon/L2Vault.sol#391-393)
maxDeposit(address) should be declared external:
        - L2Vault.maxDeposit(address) (contracts/polygon/L2Vault.sol#413-416)
maxMint(address) should be declared external:
        - L2Vault.maxMint(address) (contracts/polygon/L2Vault.sol#419-422)
maxRedeem(address) should be declared external:
        - L2Vault.maxRedeem(address) (contracts/polygon/L2Vault.sol#425-427)
maxWithdraw(address) should be declared external:
        - L2Vault.maxWithdraw(address) (contracts/polygon/L2Vault.sol#430-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

src/AffineGovernable.sol

```
AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

AffineGovernable.governance (contracts/AffineGovernable.sol#6) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

src/BaseStrategy.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.sc
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

src/BaseVault.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

src/BridgeEscrow.sol

```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

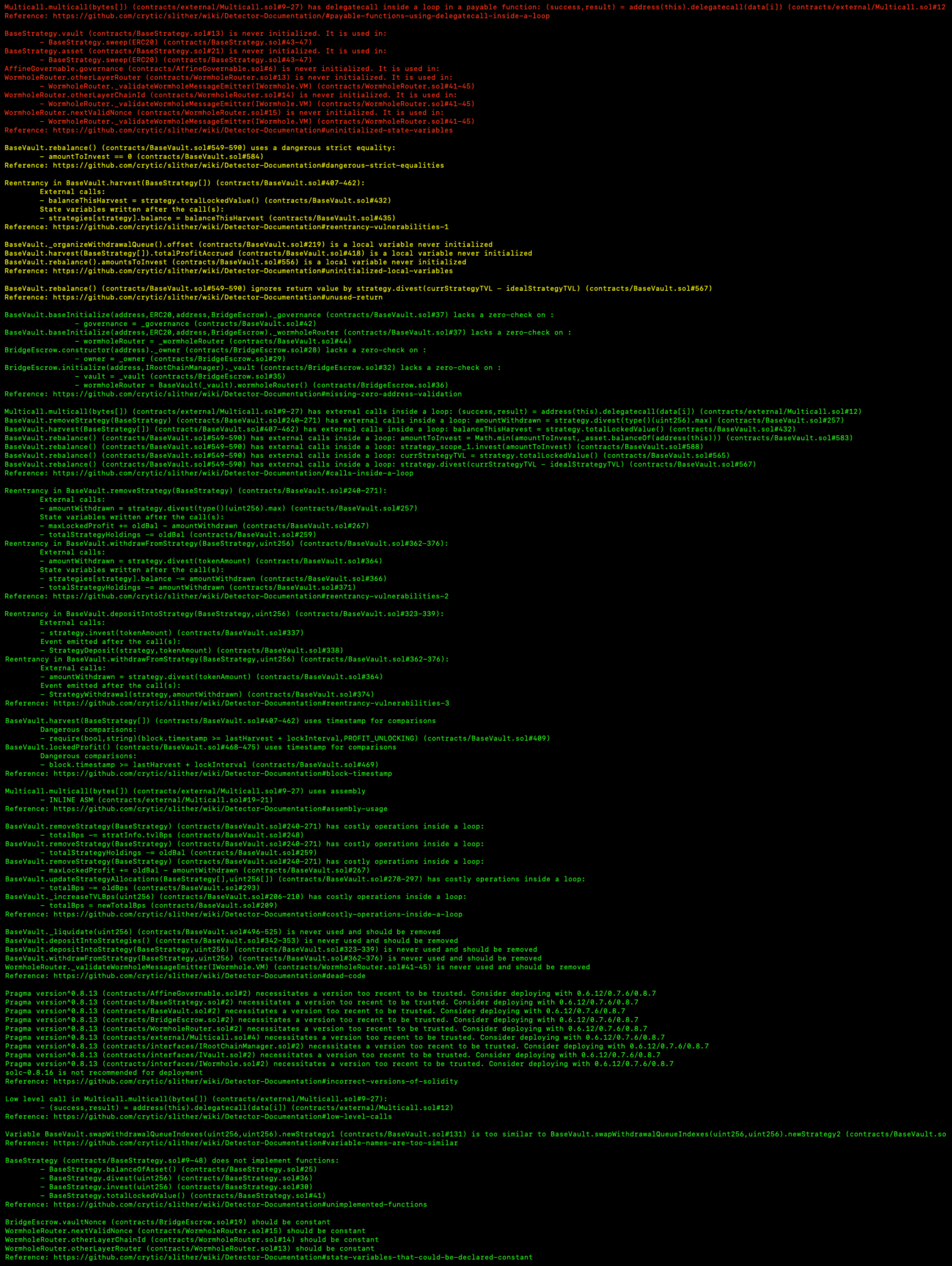
src/Constants.sol

```
Pragma version^0.8.10 (contracts/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable Constants.L1_FUND_TRANSFER_REPORT (contracts/Constants.sol#12) is too similar to Constants.L2_FUND_TRANSFER_REPORT (contracts/Constants.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

src/DollarMath.sol

```
DollarMath.add(Dollar,Dollar) (contracts/DollarMath.sol#7-9) is never used and should be removed
DollarMath.div(Dollar,Dollar) (contracts/DollarMath.sol#19-21) is never used and should be removed
DollarMath.mul(Dollar,Dollar) (contracts/DollarMath.sol#15-17) is never used and should be removed
DollarMath.sub(Dollar,Dollar) (contracts/DollarMath.sol#11-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/DollarMath.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

src/WormholeRouter.sol



```
Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

BaseStrategy.vault (contracts/BaseStrategy.sol#13) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
BaseStrategy.asset (contracts/BaseStrategy.sol#21) is never initialized. It is used in:
        - BaseStrategy.sweep(ERC20) (contracts/BaseStrategy.sol#43-47)
AffineGovernable.governance (contracts/AffineGovernable.sol#6) is never initialized. It is used in:
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) is never initialized. It is used in:
        - WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) uses a dangerous strict equality:
        - amountToInvest == 0 (contracts/BaseVault.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462):
        External calls:
        - balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
        State variables written after the call(s):
        - strategies[strategy].balance = balanceThisHarvest (contracts/BaseVault.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BaseVault._organizeWithdrawalQueue().offset (contracts/BaseVault.sol#219) is a local variable never initialized
BaseVault.harvest(BaseStrategy[]).totalProfitAccrued (contracts/BaseVault.sol#418) is a local variable never initialized
BaseVault.rebalance().amountsToInvest (contracts/BaseVault.sol#556) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BaseVault.rebalance() (contracts/BaseVault.sol#549-590) ignores return value by strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._governance (contracts/BaseVault.sol#37) lacks a zero-check on :
                - governance = _governance (contracts/BaseVault.sol#42)
BaseVault.baseInitialize(address,ERC20,address,BridgeEscrow)._wormholeRouter (contracts/BaseVault.sol#37) lacks a zero-check on :
                - wormholeRouter = _wormholeRouter (contracts/BaseVault.sol#44)
BridgeEscrow.constructor(address)._owner (contracts/BridgeEscrow.sol#28) lacks a zero-check on :
                - owner = _owner (contracts/BridgeEscrow.sol#29)
BridgeEscrow.initialize(address,IRootChainManager)._vault (contracts/BridgeEscrow.sol#32) lacks a zero-check on :
                - vault = _vault (contracts/BridgeEscrow.sol#35)
                - wormholeRouter = BaseVault(_vault).wormholeRouter() (contracts/BridgeEscrow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has external calls inside a loop: amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) has external calls inside a loop: balanceThisHarvest = strategy.totalLockedValue() (contracts/BaseVault.sol#432)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: amountToInvest = Math.min(amountToInvest,_asset.balanceOf(address(this))) (contracts/BaseVault.sol#583)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy_scope_1.invest(amountToInvest) (contracts/BaseVault.sol#588)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: currStrategyTVL = strategy.totalLockedValue() (contracts/BaseVault.sol#565)
BaseVault.rebalance() (contracts/BaseVault.sol#549-590) has external calls inside a loop: strategy.divest(currStrategyTVL - idealStrategyTVL) (contracts/BaseVault.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271):
        External calls:
        - amountWithdrawn = strategy.divest(type()(uint256).max) (contracts/BaseVault.sol#257)
        State variables written after the call(s):
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        State variables written after the call(s):
        - strategies[strategy].balance -= amountWithdrawn (contracts/BaseVault.sol#366)
        - totalStrategyHoldings -= amountWithdrawn (contracts/BaseVault.sol#371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339):
        External calls:
        - strategy.invest(tokenAmount) (contracts/BaseVault.sol#337)
        Event emitted after the call(s):
        - StrategyDeposit(strategy,tokenAmount) (contracts/BaseVault.sol#338)
Reentrancy in BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376):
        External calls:
        - amountWithdrawn = strategy.divest(tokenAmount) (contracts/BaseVault.sol#364)
        Event emitted after the call(s):
        - StrategyWithdrawal(strategy,amountWithdrawn) (contracts/BaseVault.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BaseVault.harvest(BaseStrategy[]) (contracts/BaseVault.sol#407-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastHarvest + lockInterval,PROFIT_UNLOCKING) (contracts/BaseVault.sol#409)
BaseVault.lockedProfit() (contracts/BaseVault.sol#468-475) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastHarvest + lockInterval (contracts/BaseVault.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27) uses assembly
        - INLINE ASM (contracts/external/Multicall.sol#19-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalBps -= stratInfo.tvlBps (contracts/BaseVault.sol#248)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - totalStrategyHoldings -= oldBal (contracts/BaseVault.sol#259)
BaseVault.removeStrategy(BaseStrategy) (contracts/BaseVault.sol#240-271) has costly operations inside a loop:
        - maxLockedProfit += oldBal - amountWithdrawn (contracts/BaseVault.sol#267)
BaseVault.updateStrategyAllocations(BaseStrategy[],uint256[]) (contracts/BaseVault.sol#278-297) has costly operations inside a loop:
        - totalBps -= oldBps (contracts/BaseVault.sol#293)
BaseVault._increaseTVLBps(uint256) (contracts/BaseVault.sol#206-210) has costly operations inside a loop:
        - totalBps = newTotalBps (contracts/BaseVault.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

BaseVault._liquidate(uint256) (contracts/BaseVault.sol#496-525) is never used and should be removed
BaseVault.depositIntoStrategies() (contracts/BaseVault.sol#342-353) is never used and should be removed
BaseVault.depositIntoStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#323-339) is never used and should be removed
BaseVault.withdrawFromStrategy(BaseStrategy,uint256) (contracts/BaseVault.sol#362-376) is never used and should be removed
WormholeRouter._validateWormholeMessageEmitter(IWormhole.VM) (contracts/WormholeRouter.sol#41-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (contracts/AffineGovernable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseStrategy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BaseVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/BridgeEscrow.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/WormholeRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/external/Multicall.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IRootChainManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.13 (contracts/interfaces/IWormhole.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (contracts/external/Multicall.sol#9-27):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/external/Multicall.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy1 (contracts/BaseVault.sol#131) is too similar to BaseVault.swapWithdrawalQueueIndexes(uint256,uint256).newStrategy2 (contracts/BaseVault.so
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

BaseStrategy (contracts/BaseStrategy.sol#9-48) does not implement functions:
        - BaseStrategy.balanceOfAsset() (contracts/BaseStrategy.sol#25)
        - BaseStrategy.divest(uint256) (contracts/BaseStrategy.sol#36)
        - BaseStrategy.invest(uint256) (contracts/BaseStrategy.sol#30)
        - BaseStrategy.totalLockedValue() (contracts/BaseStrategy.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BridgeEscrow.vaultNonce (contracts/BridgeEscrow.sol#19) should be constant
WormholeRouter.nextValidNonce (contracts/WormholeRouter.sol#15) should be constant
WormholeRouter.otherLayerChainId (contracts/WormholeRouter.sol#14) should be constant
WormholeRouter.otherLayerRouter (contracts/WormholeRouter.sol#13) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives. The actual vulnerabilities found by Slither are already included in the report findings.

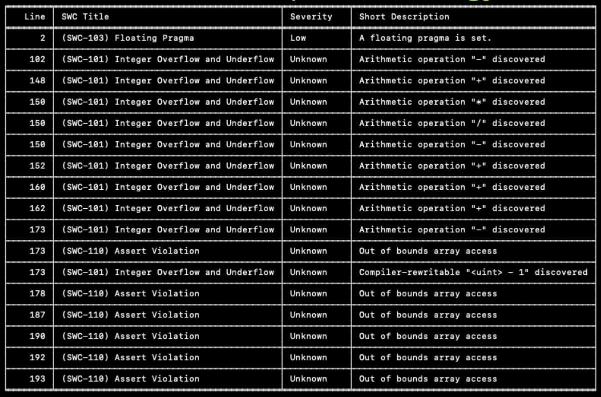
4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

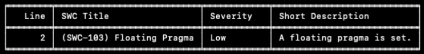
MythX results:

src/ethereum/L1CompoundStrategy.sol



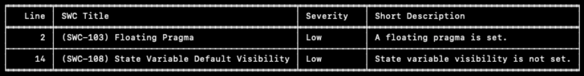
Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
102	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
148	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
150	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
150	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
150	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
152	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
160	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
162	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
173	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
173	(SWC-110) Assert Violation	Unknown	Out of bounds array access
173	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
178	(SWC-110) Assert Violation	Unknown	Out of bounds array access
187	(SWC-110) Assert Violation	Unknown	Out of bounds array access
190	(SWC-110) Assert Violation	Unknown	Out of bounds array access
192	(SWC-110) Assert Violation	Unknown	Out of bounds array access
193	(SWC-110) Assert Violation	Unknown	Out of bounds array access

src/ethereum/L1Vault.sol



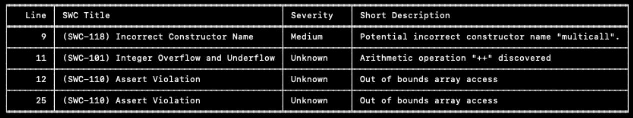
Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/ethereum/L1WormholeRouter.sol



Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
14	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

src/external/Multicall.sol



Line	SWC Title	Severity	Short Description
9	(SWC-118) Incorrect Constructor Name	Medium	Potential incorrect constructor name "multicall".
11	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
12	(SWC-110) Assert Violation	Unknown	Out of bounds array access
25	(SWC-110) Assert Violation	Unknown	Out of bounds array access

src/polygon/EmergencyWithdrawalQueue.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
18	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
61	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
61	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
71	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
73	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
74	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
83	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
84	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
93	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
99	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
104	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
105	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
115	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
118	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
119	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

src/polygon/ERC4626Router.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
7	(SWC-123) Requirement Violation	Low	Requirement violation.
55	(SWC-123) Requirement Violation	Low	Requirement violation.

src/polygon/ERC4626RouterBase.sol

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
41	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.

src/polygon/Forwarder.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/polygon/L2AAVEStrategy.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
54	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
54	(SWC-110) Assert Violation	Unknown	Out of bounds array access
54	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
114	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
156	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
158	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
158	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
158	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
160	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
171	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
182	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
182	(SWC-110) Assert Violation	Unknown	Out of bounds array access
182	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
187	(SWC-110) Assert Violation	Unknown	Out of bounds array access
198	(SWC-110) Assert Violation	Unknown	Out of bounds array access
201	(SWC-110) Assert Violation	Unknown	Out of bounds array access
203	(SWC-110) Assert Violation	Unknown	Out of bounds array access
204	(SWC-110) Assert Violation	Unknown	Out of bounds array access

src/polygon/L2Vault.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
64	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
66	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
66	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
67	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
67	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
108	(SWC-110) Assert Violation	Unknown	Out of bounds array access
109	(SWC-110) Assert Violation	Unknown	Out of bounds array access
225	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
241	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
248	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
284	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
290	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
312	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
331	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
331	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
345	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
365	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
399	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
469	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
473	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
473	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
473	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
474	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
496	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
497	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
509	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
514	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
514	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
514	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
514	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
520	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
522	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
549	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

src/polygon/L2WormholeRouter.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
14	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
51	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
64	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

src/AffineGovernable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/BaseStrategy.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/BaseVault.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
31	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

src/BridgeEscrow.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/Constants.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/DollarMath.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

src/WormholeRouter.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- No major issues found by Mythx. The floating pragma flagged by MythX is a false positive, as the pragma is set in the `hardhat.config.ts` file to the `0.8.16` version.

THANK YOU FOR CHOOSING

// HALBORN