Origami Contracts

Origami Finance

HALBORN

Origami Contracts - Origami Finance

Prepared by: HALBORN

Last Updated 12/19/2024

Date of Engagement by: December 11th, 2024 - December 16th, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
4	0	0	0	1	3

TABLE OF CONTENTS

1. Introduction

Origami Finance engaged Halborn to conduct a security assessment on their smart contracts beginning on December 11th, 2024 and ending on December 16th, 2024. The security assessment was scoped to the smart contracts provided to Halborn. Commit hashes and further details can be found in the Scope section of this report.

The Origami Finance codebase in scope mainly consists of a set of smart contracts designed to interact with the Balancer protocol, as well as Berachain's Boyco system and rewards vaults.

2. Assessment Summary

Halborn was provided 4 days for the engagement and assigned 1 full-time security engineer to review the security of the smart contracts in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were partially addressed by the Origami Finance team. The main one were the following:

- Consider transitioning critical functions to a multi-signature wallet, implementing community governance, or adding time locks to strengthen security.
- Lock the pragma version to the same version used during development and testing.
- Fix all typos to improve the readability of the codebase.

3. Test Approach And Methodology

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts 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 assessment:

- Research into architecture, purpose and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could led to arithmetic related vulnerabilities.
- Local testing with custom scripts (Foundry).
- Fork testing against main networks (Foundry).
- Static analysis of security for scoped contract, and imported functions (Slither).

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: origami

(b) Assessed Commit ID: 3182458

(c) Items in scope:

- apps/protocol/contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol
- apps/protocol/contracts/common/bera/OrigamiBeraBgtProxy.sol
- apps/protocol/contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol
- apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyManager.sol
- apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyVault.sol

Out-of-Scope: Third party dependencies and economic attacks.

REMEDIATION COMMIT ID:

- 18fb608
- d6fb329

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	3

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
CENTRALIZATION RISKS	LOW	SOLVED - 12/17/2024

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
FLOATING PRAGMA	INFORMATIONAL	NOT APPLICABLE - 12/02/2024
TYPOS	INFORMATIONAL	SOLVED - 12/18/2024
USE OF OUTDATED LIBRARIES	INFORMATIONAL	ACKNOWLEDGED - 12/17/2024

7. FINDINGS & TECH DETAILS

7.1 CENTRALIZATION RISKS

// LOW

Description

Throughout the codebase, there are some roles that control critical contract configurations and operations. These roles have unrestricted power to modify core protocol parameters, pause functionality, upgrade contracts, and control user funds with no technical restrictions or safeguards in place. If these privileged roles are compromised or act maliciously, they could manipulate protocol parameters, halt operations, or withdraw user funds, effectively enabling rug pulls.

This centralization of power directly contradicts principles of decentralization and may put user deposits at significant risk if any of the privileged roles are compromised.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (2.5)

Recommendation

Several remedial strategies can be employed, including but not limited to: transitioning control to a multi-signature wallet setup for critical functions, establishing community-driven governance for decision-making on fund management, and/or integrating time locks.

Remediation

SOLVED: The **Origami Finance team** has solved this finding, stating that:
We will have a 2/4 Origami multisig (Safe) on Berachain as owner of these contracts.

References

TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyManager.sol#L29
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyVault.sol#L21
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraBgtProxy.sol#L24
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#L18

7.2 FLOATING PRAGMA

// INFORMATIONAL

Description

All contracts in scope currently use floating pragma versions `^0.8.19` which means that the code can be compiled by any compiler version that is greater than or equal to `0.8.19`, and less than `0.9.0`.

However, it is recommended that contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively.

Score

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Lock the pragma version to the same version used during development and testing.

Remediation

NOT APPLICABLE: The **Origami Finance team** had already solved this finding in commit `18fb608` by locking the pragma in the configuration file (`foundry.toml`).

Remediation Hash

https://github.com/TempleDAO/origami/commit/18fb60820298db72e7225ee0a1c8256edca34480#diff-70ff15ee04e54068577ca63413e270d55621295c214b16dfd177e0257ec75567

References

TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyManager.sol#L1
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyVault.sol#L1
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#L1
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraBgtProxy.sol#L1
TempleDAO/origami/apps/protocol/contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#L1

7.3 TYPOS

// INFORMATIONAL

Description

Throughout the codebase, there are several instances of typos in comments. While these typos do not affect the functionality of the code, they can make the codebase harder to read and understand. It is recommended to fix these typos to improve the readability of the codebase.

Instances of this issue include:

- The word *initially* is misspelled as *initally* in the `OrigamiErc4626` contract inherited by the `OrigamiBoycoHoneyVault` contract.
- The word *receive* is misspelled as *recieve* in the `OrigamiBoycoHoneyManager` contract.
- The word *assets* is misspelled as *asests* in the `OrigamiBoycoHoneyVault` contract.

Score

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

It is recommended to fix all typos to improve the readability of the codebase.

Remediation

SOLVED: The **Origami Finance team** solved this finding in commit `d6fb329` by following the mentioned recommendation.

Remediation Hash

https://github.com/TempleDAO/origami/commit/d6fb32984436ac48884a5431791287c8e1f8b412

References

TempleDAO/origami/apps/protocol/contracts/common/OrigamiErc4626.sol#L50
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyManager.sol#L342
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyVault.sol#L90

7.4 USE OF OUTDATED LIBRARIES

// INFORMATIONAL

Description

Throughout the codebase, several OpenZeppelin contracts implementations are inherited and used. These contracts are used to implement access control, pausing functionality, upgradeable standards, and ERC4626 vaults. However, the version of these contracts are outdated (`4.9.3`) and may contain vulnerabilities that may have been fixed in newer versions (latest stable version is `5.1.0`).

For more reference about OpenZeppelin contracts versions and their vulnerabilities, see https://security.snyk.io/package/npm/@openzeppelin%2Fcontracts

BVSS

AO:A/AC:L/AX:L/R:F/S:U/C:N/A:N/I:M/D:M/Y:M (1.9)

Recommendation

Consider updating all OpenZeppelin contracts to the latest versions to benefit from the latest security patches and improvements.

Remediation

ACKNOWLEDGED: The **Origami Finance team** made a business decision to acknowledge this finding and not alter the contracts, stating:
We might create a future PR to update deps to a newer version.

References

TempleDAO/origami/apps/protocol/contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#L5-L7
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraBgtProxy.sol#L5-L8
TempleDAO/origami/apps/protocol/contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#L5-L6
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyManager.sol#L5-L8
TempleDAO/origami/apps/protocol/contracts/investments/bera/OrigamiBoycoHoneyVault.sol#L5-L7

STATIC ANALYSIS REPORT

Description

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the 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.

The security team assessed all findings identified by the Slither software, however, findings with related to external dependencies are not included in the below results for the sake of report readability.

Output

The findings obtained as a result of the Slither scan were reviewed, and the majority were not included in the report because they were determined as false positives.

```
INFO:Detectors:
OrigamiErc4626._depositHook(address,uint256) (contracts/common/OrigamiErc4626.sol#450-453) uses arbitrary from in transferFrom: SafeERC20.safeTransferFrom(_asset,caller,address(this),assets) (contracts/common/OrigamiErc4626.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
OrigamiBoycoHoneyManager.constructor(address,address,address,address,address,address,address,address)._honeyIndex (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#99) is a local variable never initialized
OrigamiBoycoHoneyManager.constructor(address,address,address,address,address,address,address,address)._usdcIndex (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#98) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
OrigamiBalancerComposableStablePoolHelper.constructor(address,address,bytes32) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#54-73) ignores return value by (lpTokenAddr,None) = balancerVault.getPool(poolId) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#59)
OrigamiBalancerComposableStablePoolHelper.constructor(address,address,bytes32) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#54-73) ignores return value by (addresses,None,None) = balancerVault.getPoolTokens(poolId) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#62)
OrigamiBalancerComposableStablePoolHelper.getTokenBalances() (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#84-86) ignores return value by (None,balances,None) = balancerVault.getPoolTokens(poolId) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#85)
OrigamiBalancerComposableStablePoolHelper.proportionalAddLiquidityQuote(uint256,uint256,uint256) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#133-194) ignores return value by (expectedLpTokenAmount,None) = balancerQueries.queryJoin({poolId:poolId,sender:address(0),recipient:address(0),request:requestData}) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#181-186)
OrigamiBalancerComposableStablePoolHelper.proportionalRemoveLiquidityQuote(uint256,uint256) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#229-265) ignores return value by (None,expectedTokenAmounts) = balancerQueries.queryExit(poolId,address(0),address(0),requestData) (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#250-255)
OrigamiBeraRewardsVaultProxy.getReward(address) (contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#87-89) ignores return value by rewardsVault.getReward(address(this),recipient) (contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#88)
OrigamiBoycoHoneyManager.deployUsdcInputs(uint256,uint256) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#159-220) ignores return value by (tokenAmounts,None,None,requestData) = bexPoolHelper.proportionalAddLiquidityQuote(BEX_TOKEN_INDEX_HONEY_,honeyToPair,slippageBps) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#212-216)
OrigamiBoycoHoneyManager.recallUsdcInputs(uint256,uint256) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#248-250) ignores return value by (None,None,requestData) = bexPoolHelper.proportionalRemoveLiquidityQuote(bexLpAmount,slippageBps) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#249)
OrigamiErc4626._maxWithdraw(address,uint256) (contracts/common/OrigamiErc4626.sol#370-377) ignores return value by (shares,None) = shares.splitSubtractBps(feeBps,OrigamiMath.Rounding.ROUND_DOWN) (contracts/common/OrigamiErc4626.sol#375)
OrigamiBoycoHoneyVault._depositHook(address,uint256) (contracts/investments/bera/OrigamiBoycoHoneyVault.sol#81-84) ignores return value by _manager.deposit(assets) (contracts/investments/bera/OrigamiBoycoHoneyVault.sol#83)
OrigamiBoycoHoneyVault._withdrawHook(uint256,address) (contracts/investments/bera/OrigamiBoycoHoneyVault.sol#87-89) ignores return value by _manager.withdraw(assets,receiver) (contracts/investments/bera/OrigamiBoycoHoneyVault.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
OrigamiErc4626.maxMint(address)._totalSupply (contracts/common/OrigamiErc4626.sol#139) shadows:
        - ERC20._totalSupply (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#43) (state variable)
OrigamiErc4626._maxDeposit(uint256)._totalSupply (contracts/common/OrigamiErc4626.sol#353) shadows:
        - ERC20._totalSupply (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#43) (state variable)
IOrigamiErc4626.setMaxTotalSupply(uint256).maxTotalSupply (contracts/interfaces/common/IOrigamiErc4626.sol#65) shadows:
        - IOrigamiErc4626.maxTotalSupply() (contracts/interfaces/common/IOrigamiErc4626.sol#83) (function)
IOrigamiErc4626.seedDeposit(uint256,address,uint256).maxTotalSupply (contracts/interfaces/common/IOrigamiErc4626.sol#73) shadows:
        - IOrigamiErc4626.maxTotalSupply() (contracts/interfaces/common/IOrigamiErc4626.sol#83) (function)
IOrigamiBoycoHoneyManager.setBeraRewardsVaultProxy(address).beraRewardsVaultProxy (contracts/interfaces/investments/bera/IOrigamiBoycoHoneyManager.sol#31) shadows:
        - IOrigamiBoycoHoneyManager.beraRewardsVaultProxy() (contracts/interfaces/investments/bera/IOrigamiBoycoHoneyManager.sol#99) (function)
IOrigamiBoycoHoneyManager.setBexPoolHelper(address).bexPoolHelper (contracts/interfaces/investments/bera/IOrigamiBoycoHoneyManager.sol#30) shadows:
        - IOrigamiBoycoHoneyManager.bexPoolHelper() (contracts/interfaces/investments/bera/IOrigamiBoycoHoneyManager.sol#89) (function)
IOrigamiDelegated4626Vault.setTokenPrices(address).tokenPrices (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626Vault.sol#21) shadows:
        - IOrigamiDelegated4626Vault.tokenPrices() (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626Vault.sol#38) (function)
IOrigamiDelegated4626Vault.setManager(address).manager (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626Vault.sol#26) shadows:
        - IOrigamiDelegated4626Vault.manager() (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626Vault.sol#44) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in OrigamiErc4626._deposit(address,address,uint256,uint256) (contracts/common/OrigamiErc4626.sol#439-445):
        External calls:
        - _depositHook(caller,assets) (contracts/common/OrigamiErc4626.sol#440)
                - SafeERC20.safeTransferFrom(_asset,caller,address(this),assets) (contracts/common/OrigamiErc4626.sol#452)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#122)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#135)
        External calls sending eth:
        - _depositHook(caller,assets) (contracts/common/OrigamiErc4626.sol#440)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#135)
        State variables written after the call(s):
        - _mint(receiver,shares) (contracts/common/OrigamiErc4626.sol#442)
                - _balances[account] += amount (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#259)
        - _mint(receiver,shares) (contracts/common/OrigamiErc4626.sol#442)
                - _totalSupply += amount (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#256)
Reentrancy in OrigamiBoycoHoneyManager.withdraw(uint256,address) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#144-156):
        External calls:
        - usdcToken.safeTransfer(receiver,assetsWithdrawn) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#154)
        State variables written after the call(s):
        - totalAssets -= assetsWithdrawn (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#155)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in OrigamiErc4626._deposit(address,address,uint256,uint256) (contracts/common/OrigamiErc4626.sol#439-445):
        External calls:
        - _depositHook(caller,assets) (contracts/common/OrigamiErc4626.sol#440)
                - SafeERC20.safeTransferFrom(_asset,caller,address(this),assets) (contracts/common/OrigamiErc4626.sol#452)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#122)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#135)
        External calls sending eth:
        - _depositHook(caller,assets) (contracts/common/OrigamiErc4626.sol#440)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#135)
        Event emitted after the call(s):
        - Deposit(caller,receiver,assets,shares) (contracts/common/OrigamiErc4626.sol#444)
        - Transfer(address(0),account,amount) (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#261)
                - _mint(receiver,shares) (contracts/common/OrigamiErc4626.sol#442)
Reentrancy in OrigamiBoycoHoneyManager.deployUsdc(uint256,uint256,IBalancerVault.JoinPoolRequest) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#223-245):
        External calls:
        - usdcToken.forceApprove(address(honeyFactory),usdcAmountToSell) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#231)
        - honeyReceived = honeyFactory.mint(address(usdcToken),usdcAmountToSell,address(this)) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#232)
        - usdcToken.forceApprove(address(bexPoolHelper),usdcAmountToPair) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#235)
        - honeyToken.forceApprove(address(bexPoolHelper),honeyReceived) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#236)
        - bexPoolHelper.addLiquidity(address(this),requestData) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#237)
        - bexLpToken.safeTransfer(address(beraRewardsVaultProxy),lpBalance) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#241)
        - beraRewardsVaultProxy.stake(lpBalance) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#242)
        Event emitted after the call(s):
        - UsdcDeployed(usdcAmountToPair,usdcAmountToSell,honeyReceived,lpBalance) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#244)
Reentrancy in OrigamiBoycoHoneyManager.recallUsdc(uint256,IBalancerVault.ExitPoolRequest) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#253-274):
        External calls:
        - beraRewardsVaultProxy.withdraw(lpTokenAmount,address(this)) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#259)
        - bexLpToken.forceApprove(address(bexPoolHelper),lpTokenAmount) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#262)
        - bexPoolHelper.removeLiquidity(lpTokenAmount,address(this),requestData) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#263)
        - honeyToken.forceApprove(address(honeyFactory),honeyBalance) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#268)
        - usdcFromRedeem = honeyFactory.redeem(address(usdcToken),honeyBalance,address(this))[0] (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#271)
        Event emitted after the call(s):
        - UsdcRecalled(usdcFromLp,usdcFromRedeem,honeyBalance,lpTokenAmount) (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
OrigamiErc4626.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/common/OrigamiErc4626.sol#285-300) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > deadline (contracts/common/OrigamiErc4626.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
OrigamiErc4626._depositHook(address,uint256) (contracts/common/OrigamiErc4626.sol#450-453) is never used and should be removed
OrigamiErc4626._withdrawHook(uint256,address) (contracts/common/OrigamiErc4626.sol#482-485) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Version constraint ^0.8.19 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess.
It is used by:
        - ^0.8.19 (contracts/common/MintableToken.sol#1)
        - ^0.8.19 (contracts/common/OrigamiErc4626.sol#1)
        - ^0.8.19 (contracts/common/TokenPrices.sol#1)
        - ^0.8.19 (contracts/common/access/OrigamiElevatedAccess.sol#1)
        - ^0.8.19 (contracts/common/access/OrigamiElevatedAccessBase.sol#1)
        - ^0.8.19 (contracts/common/access/OrigamiElevatedAccessUpgradeable.sol#1)
        - ^0.8.19 (contracts/common/balancer/OrigamiBalancerComposableStablePoolHelper.sol#1)
        - ^0.8.19 (contracts/common/bera/OrigamiBeraBgtProxy.sol#1)
        - ^0.8.19 (contracts/common/bera/OrigamiBeraRewardsVaultProxy.sol#1)
        - ^0.8.19 (contracts/interfaces/investments/bera/IOrigamiBoycoHoneyManager.sol#1)
        - ^0.8.19 (contracts/investments/bera/OrigamiBoycoHoneyManager.sol#1)
```

```
        - ^0.8.19 (contracts/investments/bera/OrigamiBoycoHoneyVault.sol#1)
        - ^0.8.19 (contracts/investments/util/OrigamiManagerPausable.sol#1)
        - ^0.8.19 (contracts/libraries/CommonEventsAndErrors.sol#1)
        - ^0.8.19 (contracts/libraries/DynamicFees.sol#1)
        - ^0.8.19 (contracts/libraries/OrigamiMath.sol#1)
Version constraint ^0.8.4 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables.
It is used by:
        - ^0.8.4 (contracts/interfaces/common/IMintableToken.sol#1)
        - ^0.8.4 (contracts/interfaces/common/IOrigamiErc4626.sol#1)
        - ^0.8.4 (contracts/interfaces/common/IRepricingToken.sol#1)
        - ^0.8.4 (contracts/interfaces/common/ITokenPrices.sol#1)
        - ^0.8.4 (contracts/interfaces/common/IWrappedToken.sol#1)
        - ^0.8.4 (contracts/interfaces/common/access/IOrigamiElevatedAccess.sol#1)
        - ^0.8.4 (contracts/interfaces/common/access/IWhitelisted.sol#1)
        - ^0.8.4 (contracts/interfaces/common/balancer/IOrigamiBalancerPoolHelper.sol#1)
        - ^0.8.4 (contracts/interfaces/common/bera/IOrigamiBeraBgtProxy.sol#1)
        - ^0.8.4 (contracts/interfaces/common/bera/IOrigamiBeraRewardsVaultProxy.sol#1)
        - ^0.8.4 (contracts/interfaces/common/borrowAndLend/IOrigamiBorrowAndLend.sol#1)
        - ^0.8.4 (contracts/interfaces/common/circuitBreaker/IOrigamiCircuitBreaker.sol#1)
        - ^0.8.4 (contracts/interfaces/common/circuitBreaker/IOrigamiCircuitBreakerProxy.sol#1)
        - ^0.8.4 (contracts/interfaces/common/flashLoan/IOrigamiFlashLoanProvider.sol#1)
        - ^0.8.4 (contracts/interfaces/common/flashLoan/IOrigamiFlashLoanReceiver.sol#1)
        - ^0.8.4 (contracts/interfaces/common/interestRate/IInterestRateModel.sol#1)
        - ^0.8.4 (contracts/interfaces/common/oracle/IOrigamiOracle.sol#1)
        - ^0.8.4 (contracts/interfaces/external/balancer/IBalancerBptToken.sol#1)
        - ^0.8.4 (contracts/interfaces/external/balancer/IBalancerQueries.sol#1)
        - ^0.8.4 (contracts/interfaces/external/balancer/IBalancerVault.sol#1)
        - ^0.8.4 (contracts/interfaces/external/bera/IBeraBgt.sol#2)
        - ^0.8.4 (contracts/interfaces/external/bera/IBeraHoneyFactory.sol#2)
        - ^0.8.4 (contracts/interfaces/external/bera/IBeraHoneyFactoryReader.sol#2)
        - ^0.8.4 (contracts/interfaces/external/bera/IBeraRewardsVault.sol#1)
        - ^0.8.4 (contracts/interfaces/external/chainlink/IAggregatorV3Interface.sol#1)
        - ^0.8.4 (contracts/interfaces/external/chainlink/IKeeperCompatibleInterface.sol#1)
        - ^0.8.4 (contracts/interfaces/external/lido/IStETH.sol#1)
        - ^0.8.4 (contracts/interfaces/external/lido/IWstETH.sol#1)
        - ^0.8.4 (contracts/interfaces/external/traderJoe/IJoeLBQuoter.sol#2)
        - ^0.8.4 (contracts/interfaces/external/uniswap/IUniswapV3Pool.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiInvestment.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiInvestmentManager.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiInvestmentVault.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiOToken.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiOTokenManager.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/IOrigamiOTokenManagerWithNative.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626Vault.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/erc4626/IOrigamiDelegated4626VaultManager.sol#1)
        - ^0.8.4 (contracts/interfaces/investments/util/IOrigamiManagerPausable.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Slither:. analyzed (89 contracts with 94 detectors), 32 result(s) found
```

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the

project's integrity and addressing potential vulnerabilities introduced by code modifications.