inception^{LRT}

INCEPTIOLRT SYMBIOTIC VAULT SMART CONTRACTS SECURITY AUDIT REPORT

OXORIO

MARCH 7, 2025

1 EXECUTIVE SUMMARY

OXORIO

1.1 EXECUTIVE SUMMARY

This document presents the smart contracts security audit conducted by Oxorio for Inception's InceptioLRT Symbiotic Vault.

`InceptionLRT` is the Modular Aggregation Layer for Restaking, designed to address the growing complexity and fragmentation of the restaking ecosystem. By uniting diverse strategies, assets, and protocols under one roof, `InceptionLRT` empowers users to take full control of their restaking journey while unlocking unprecedented opportunities in DeFi.
Restaking offers limitless potential for yield generation, with countless ways to secure networks and earn rewards. However, the ecosystem is fragmented, making it difficult for users to navigate, optimize returns, and minimize risks. `InceptionLRT` bridges this gap by delivering a seamless, modular platform that simplifies the restaking process while maximizing efficiency and accessibility.

The `InceptionLRT` Symbiotic Vault project is an innovative solution for restaking in the Ethereum ecosystem, enabling users to participate in restaking through the Symbiotic protocol using the Mellow Protocol infrastructure.

The audit process involved a comprehensive approach, including manual code review, automated analysis, and extensive testing and simulations of the smart contracts to assess the project's security and functionality. The audit covered a total of 29 smart contracts, encompassing 2013 lines of code. The codebase was thoroughly examined, with the audit team collaborating closely with Inception and referencing the provided documentation to address any questions regarding the expected behavior. For an in-depth explanation of used the smart contract security audit methodology, please refer to the Security Assessment Methodology section of this document.

Throughout the audit, a collaborative approach was maintained with Inception to address all concerns identified within the audit's scope. Each issue has been either resolved or formally acknowledged by Inception, contributing to the robustness of the project.

As a result, following a comprehensive review, our auditors have verified that the InceptionLRT, as of audited commit `bfae89718774a612ba713f23519a984389eeefaa`, has met the security and functionality requirements established for this audit, based on the code and documentation provided, and operates as intended within the defined scope.

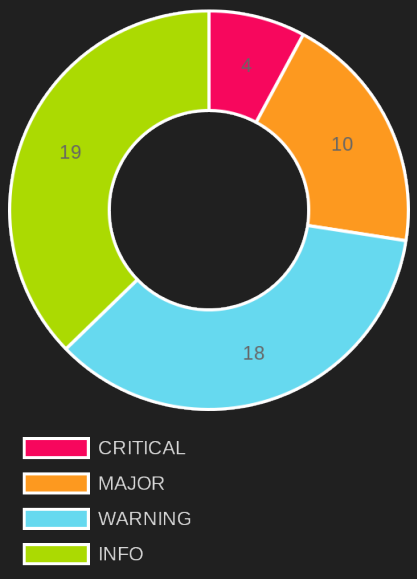
1.2 SUMMARY OF FINDINGS

The table below provides a comprehensive summary of the audit findings, categorizing each by status and severity level. For a detailed description of the severity levels and statuses of findings, see the Findings Classification Reference section.

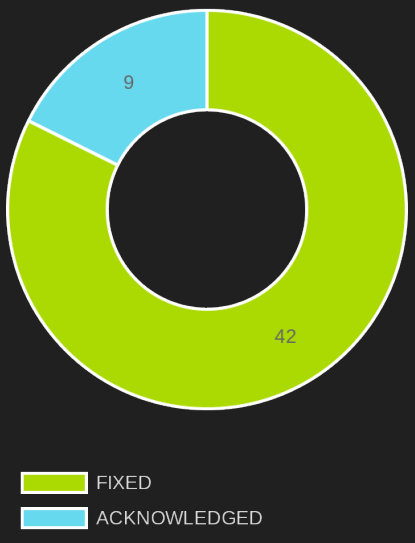
Detailed technical information on the audit findings, along with our recommendations for addressing them, is provided in the Findings Report section for further reference.

All identified issues have been addressed, with Inception fixing them or formally acknowledging their status.

Severity	TOTAL	NEW	FIXED	ACKNOWLEDGED	NO ISSUE
CRITICAL	**4**	0	4	0	0
MAJOR	**10**	0	10	0	0
WARNING	**18**	0	14	4	0
INFO	**19**	0	14	5	0
TOTAL	**51**	**0**	**42**	**9**	**0**



Issue distribution by severity



Issue distribution by status

2 AUDIT OVERVIEW

CONTENTS

2.1 DISCLAIMER

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

This report is based on the scope of materials and documentation provided to Oxorio for the security audit as detailed in the Executive Summary and Audited Files sections. The findings presented in this report may not encompass all potential vulnerabilities. Oxorio delivers this report and its findings on an as-is basis, and any reliance on this report is undertaken at the user's sole risk. It is important to recognize that blockchain technology remains in a developmental stage and is subject to inherent risks and flaws.

This audit does not extend beyond the programming language of smart contracts to include areas such as the compiler layer or other components that may introduce security risks. Consequently, this report should not be interpreted as an endorsement of any project or team, nor does it guarantee the security of the project under review.

THE CONTENT OF THIS REPORT, INCLUDING ITS ACCESS AND/OR USE, AS WELL AS ANY ASSOCIATED SERVICES OR MATERIALS, MUST NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE. Third parties should not rely on this report for making any decisions, including the purchase or sale of any product, service, or asset. Oxorio expressly disclaims any liability related to the report, its contents, and any associated services, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Oxorio does not warrant, endorse, or take responsibility for any product or service referenced or linked within this report.

For any decisions related to financial, legal, regulatory, or other professional advice, users are strongly encouraged to consult with qualified professionals.

2.2 PROJECT BRIEF

Title	Description
Client	Inception
Project name	InceptionLRT Symbiotic Vault
Category	Liquid Restaking
Website	inceptionlrt.com
Documentation	github.com/inceptionlrt/smart-contracts/tree/6d2576a938d20cf37efe62a79d25a90f7d67634c/projects/vaults
Repository	github.com/inceptionlrt/smart-contracts
Initial Commit	`6d2576a938d20cf37efe62a79d25a90f7d67634c`
Reaudited Commit 1	`f86701e8007c2bec11d95b4ee419863bcd801cba`
Reaudited Commit 2	`cf7ab740aebbed70a828f4a76bd763ef46bd0a25`
Reaudited Commit 3	`bfae89718774a612ba713f23519a984389eeefa`
Platform	L1
Network	Ethereum
Languages	Solidity
Lead Auditor	Alexander Mazaletskiy - am@oxor.io
Project Manager	Elena Kozmiryuk - elenak@oxor.io

2.3 PROJECT TIMELINE

The key events and milestones of the project are outlined below.

Date	Event
December 25, 2024	Client engaged Oxorio requesting an audit.
January 15, 2025	The audit team initiated work on the project.
January 23, 2025	Submission of the comprehensive audit report #1.
February 3, 2025	Client's feedback on the audit report was received.
February 3, 2025	The audit team commenced work on a re-audit of the project.
Februrary 6, 2025	Submission of the comprehensive audit report #2.
February 24, 2025	The audit team commenced work on a re-audit of the project.
Februrary 25, 2025	Submission of the comprehensive audit report #3.
February 26, 2025	The audit team commenced work on a re-audit of the project.
March 7, 2025	Submission of the final comprehensive audit.

2.4 AUDITED FILES

The following table contains a list of the audited files. The scc tool was used to count the number of lines and assess complexity of the files.

	File	Lines	Blanks	Comments	Code	Complexity
1	projects/vaults/contracts/assets-handler/InceptionAssetsHandler.sol	67	13	9	**45**	2
2	projects/vaults/contracts/interfaces/common/IInceptionRatioFeed.sol	6	1	1	**4**	0
3	projects/vaults/contracts/interfaces/common/IInceptionToken.sol	13	4	1	**8**	0
4	projects/vaults/contracts/interfaces/common/IInceptionVaultErrors.sol	64	30	3	**31**	0
5	projects/vaults/contracts/interfaces/common/IOwnable.sol	6	1	1	**4**	0
6	projects/vaults/contracts/interfaces/common/IRateProvider.sol	9	1	4	**4**	0
7	projects/vaults/contracts/interfaces/common/IrEth.sol	11	3	2	**6**	0
8	projects/vaults/contracts/interfaces/common/IStEth.sol	16	4	1	**11**	0
9	projects/vaults/contracts/interfaces/symbiotic-vault/IIMellowRestaker.sol	100	37	5	**58**	0
10	projects/vaults/contracts/interfaces/symbiotic-vault/IInceptionVault_S.sol	83	24	4	**55**	0
11	projects/vaults/contracts/interfaces/symbiotic-vault/IMellowHandler.sol	39	7	4	**28**	0
12	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IDefaultBondStrategy.sol	81	10	48	**23**	0
13	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowDefaultCollateral.sol	38	3	22	**13**	0
14	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowDepositWrapper.sol	76	9	39	**28**	0
15	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowHandler.sol	12	4	1	**7**	0
16	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowPriceOracle.sol	21	1	13	**7**	0
17	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowRatiosOracle.sol	21	1	13	**7**	0
18	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowVault.sol	407	43	194	**170**	0
19	projects/vaults/contracts/interfaces/symbiotic-vault/mellow-core/IMellowVaultConfigurator.sol	392	98	178	**116**	0
20	projects/vaults/contracts/interfaces/symbiotic-vault/symbiotic-core/ISymbioticVault.sol	261	27	131	**103**	0
21	projects/vaults/contracts/lib/Convert.sol	60	4	9	**47**	17
22	projects/vaults/contracts/lib/FullMath.sol	131	10	54	**67**	18
23	projects/vaults/contracts/lib/InceptionLibrary.sol	81	8	13	**60**	25
24	projects/vaults/contracts/mellow-handler/MellowHandler.sol	210	36	51	**123**	10
25	projects/vaults/contracts/restakers/IMellowRestaker.sol	422	69	31	**322**	18
26	projects/vaults/contracts/tokens/InceptionToken.sol	105	18	19	**68**	1
27	projects/vaults/contracts/vaults/Symbiotic/InceptionVault_S.sol	607	105	78	**424**	14
28	projects/vaults/contracts/vaults/Symbiotic/vault_e2/InVault_S_E2.sol	41	5	3	**33**	0

	File	Lines	Blanks	Comments	Code	Complexity
29	projects/vaults/contracts/restakers/ISymbioticRestaker.sol	196	36	19	**141**	15
	Total	**3576**	**612**	**951**	**2013**	**24**

Lines: The total number of lines in each file. This provides a quick overview of the file size and its contents.

Blanks: The count of blank lines in the file.

Comments: This column shows the number of lines that are comments.

Code: The count of lines that actually contain executable code. This metric is essential for understanding how much of the file is dedicated to operational elements rather than comments or whitespace.

Complexity: This column shows the file complexity per line of code. It is calculated by dividing the file's total complexity (an approximation of cyclomatic complexity that estimates logical depth and decision points like loops and conditional branches) by the number of executable lines of code. A higher value suggests greater complexity per line, indicating areas with concentrated logic.

2.5 PROJECT OVERVIEW

`InceptionLRT` is the Modular Aggregation Layer for Restaking, designed to address the growing complexity and fragmentation of the restaking ecosystem. By uniting diverse strategies, assets, and protocols under one roof, `InceptionLRT` empowers users to take full control of their restaking journey while unlocking unprecedented opportunities in DeFi.
Restaking offers limitless potential for yield generation, with countless ways to secure networks and earn rewards. However, the ecosystem is fragmented, making it difficult for users to navigate, optimize returns, and minimize risks. `InceptionLRT` bridges this gap by delivering a seamless, modular platform that simplifies the restaking process while maximizing efficiency and accessibility.

The `InceptionLRT` Symbiotic Vault project is an innovative solution for restaking in the Ethereum ecosystem, enabling users to participate in restaking through the Symbiotic protocol using the Mellow Protocol infrastructure.

2.6 CODEBASE QUALITY ASSESSMENT

The Codebase Quality Assessment table offers a comprehensive assessment of various code metrics, as evaluated by our team during the audit, to gauge the overall quality and maturity of the project's codebase. By evaluating factors such as complexity, documentation and testing coverage to best practices, this table highlights areas where the project excels and identifies potential improvement opportunities. Each metric receives an individual rating, offering a clear snapshot of the project's current state, guiding prioritization for refactoring efforts, and providing insights into its maintainability, security, and scalability. For a detailed description of the categories and ratings, see the Codebase Quality Assessment Reference section.

Category	Assessment	Result
Access Control	During the initial review, the protocol implemented a system of access control separation between the owner, operator, and trusteeManager. However, issue `M-02` was identified as requiring attention. This issue was successfully identified and resolved during the re-audit process.	**Excellent**
Arithmetic	The protocol relies heavily on a significant number of arithmetic operations, which necessitates careful consideration of potential overflow and underflow risks. During the initial review, the following issues were identified as areas of concern: `C-02`, `M-05`, and `W-01`. These issues were subsequently addressed and resolved during the re-audit process.	**Good**
Complexity	The protocol features a well-defined modular architecture, which enhances readability, maintainability, and scalability. However, there are instances of code duplication that could be optimized to improve efficiency and reduce potential risks.	**Good**
Data Validation	The protocol implements input validation in certain areas; however, previously, there were multiple instances where validation was lacking, which could have led to vulnerabilities, unexpected behavior, or exploitation. During the re-audit, these issues (`W-04`, `W-06`, `W-07`) were reviewed and resolved. The fixes ensure that input validation is now more consistent and robust across the protocol, mitigating the risks associated with insufficient validation and improving the overall security and reliability of the system.	**Good**
Decentralization	The protocol is not decentralized.	**Not Applicable**

Category	Assessment	Result
Documentation	The documentation provides a high-level overview of how the protocol operates, but it lacks detailed descriptions of the architecture and processes that occur within the protocol.	**Poor**
External Dependencies	The protocol successfully integrates with Mellow and Symbiotic Protocols, demonstrating effective interoperability and functionality. Previously, critical issues `C-01` and `C-03` were identified, requiring attention to ensure the security, efficiency, and reliability of the integration. However, these issues have been successfully resolved during the re-audit, significantly enhancing the overall stability and security of the system.	**Good**
Error Handling	The protocol demonstrates robust error handling and effectively utilizes custom errors.	**Excellent**
Logging and Monitoring	The protocol incorporates a comprehensive set of events designed to monitor and track operations, which is crucial for transparency, debugging, and off-chain analytics. Previously, a specific issue, `W-05`, was identified in this area. However, this issue has been successfully addressed, reinforcing the protocol's effectiveness in ensuring transparency and enabling robust operational monitoring.	**Excellent**
Low-Level Calls	The protocol avoids the use of low-level operations (e.g., call, delegatecall, staticcall, and assembly (inline assembly)), which is generally a positive design choice as it reduces complexity and minimizes potential vulnerabilities.	**Not Applicable**
Testing and Verification	Despite having a significant number of tests, the protocol does not cover certain critical cases and has several high-risk areas that remain untested or inadequately tested. This lack of comprehensive test coverage increases the risk of vulnerabilities and unexpected behavior in production.	**Fair**

2.7 FINDINGS BREAKDOWN BY FILE

This table provides an overview of the findings across the audited files, categorized by severity level. It serves as a useful tool for identifying areas that may require attention, helping to prioritize remediation efforts, and provides a clear summary of the audit results.

File	TOTAL	CRITICAL	MAJOR	WARNING	INFO
/projects/vaults/contracts/restakers/ISymbioticRestaker.sol	**12**	0	1	5	6
projects/vaults/contracts/vaults/Symbiotic/InceptionVault_S.sol	**12**	1	2	3	6
projects/vaults/contracts/restakers/IMellowRestaker.sol	**10**	3	2	2	3
projects/vaults/contracts/mellow-handler/MellowHandler.sol	**5**	0	1	2	2
/projects/vaults/contracts/symbiotic-handler/SymbioticHandler.sol	**4**	0	1	1	2
/projects/vaults/contracts/restakers/IMellowRestaker.sol	**3**	0	1	1	1
/projects/vaults/contracts/vaults/Symbiotic/InceptionVault_S.sol	**3**	0	1	0	2
projects/vaults/contracts/tokens/InceptionToken.sol	**3**	0	0	2	1
/projects/vaults/contracts/restakers/IBaseRestaker.sol3	**1**	0	0	0	1
/projects/vaults/contracts/vaults/InceptionBasicStrategyVault.sol	**1**	0	0	0	1
projects/restaking-pool/contracts/libraries/InceptionLibrary.sol	**1**	0	0	1	0
projects/vaults/contracts/lib/InceptionLibrary.sol	**1**	0	0	1	0
projects/vaults/contracts/restakers/ISymbioticRestaker.sol	**1**	0	1	0	0
projects/vaults/contracts/symbiotic-handler/SymbioticHandler.sol	**1**	0	0	1	0

2.8 CONCLUSION

A comprehensive audit was conducted on 29 smart contracts, revealing 4 critical and 10 major issues, along with numerous warnings and informational notes.

The audit highlighted various attack vectors and potential vulnerabilities, with significant findings related to issues such as unnecessary computations (e.g., 1 wei addition) and inefficient loops were identified, which could lead to gas inefficiencies and potential Denial of Service (DoS) risks, missing or insufficient validation checks for critical parameters (e.g., ratio, `maxFlashFeeRate`, `optimalWithdrawalRate`) and improper handling of returned tokens were noted, which could lead to unauthorized access or imbalances in the system, risks associated with external dependencies, such as ratioFeed, were identified, where compromised or incorrect values could lead to vulnerabilities.
These findings underscore the need for improved validation, enhanced access controls, and optimized code to mitigate risks and ensure the system's security and reliability. Addressing these issues is critical to safeguarding the contracts against potential attacks and ensuring compliance with industry best practices.

The proposed changes are aimed at addressing critical and major issues, including preventing overflows, division-by-zero errors, and invalid parameter settings, while ensuring proper validation of critical values and enhancing the overall robustness of the smart contracts. These recommendations focus on improving parameter validation, removing unnecessary computations, ensuring accurate error handling, and reinforcing role management integrity to strengthen the security and reliability of the system. Based on industry best practices, these enhancements will mitigate potential risks, improve code quality, and ensure the contracts meet the highest security standards. We strongly advise addressing the identified issues to safeguard the system against vulnerabilities, improve the quality of the codebase, and ensure its long-term stability.

Moreover, we advise increasing the test coverage of the codebase. Comprehensive testing is essential to uncover edge cases and ensure that the smart contracts perform as expected under various conditions. Enhancing test coverage will not only improve the reliability of the contracts but also contribute to the overall security of the project.

As stated in each particular issue, each critical, major, and warning issue identified has been correctly fixed or acknowledged by the client, so contracts are assumed as secure to use according to our security criteria. Final commit identifier with all fixes: `bfae89718774a612ba713f23519a984389eeefaa`. This version is recommended to deploy to testnet for further system testing.

To further help the project reach a production-ready state, we highly advise additional rounds of security reviews after every change in contracts.

3 FINDINGS REPORT

OXORIO

3.1 CRITICAL

C-01	Incorrect calculation of `lpAmountToAmount` in `IMellowRestaker`
Severity	**CRITICAL**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `lpAmountToAmount`	372

Description

In function `lpAmountToAmount` of contract `IMellowRestaker`, there is an incorrect calculation of `lpAmountToAmount`. When referring to the Mellow protocol, it can be observed that the calculation of `expectedAmount`, which is analogous to `lpAmountToAmount`, occurs in the `analyzeRequest` function:

```
function analyzeRequest(
    ProcessWithdrawalsStack memory s,
    WithdrawalRequest memory request
) public pure returns (bool, bool, uint256[] memory expectedAmounts) {
    uint256 lpAmount = request.lpAmount;
    if (
        request.tokensHash != s.tokensHash || request.deadline < s.timestamp
    ) return (false, false, expectedAmounts);

    uint256 value = FullMath.mulDiv(lpAmount, s.totalValue, s.totalSupply);
    value = FullMath.mulDiv(value, D9 - s.feeD9, D9);
    uint256 coefficientX96 = FullMath.mulDiv(value, Q96, s.ratiosX96Value);

    uint256 length = s.erc20Balances.length;
    expectedAmounts = new uint256[](length);
    for (uint256 i = 0; i < length; i++) {
        uint256 ratiosX96 = s.ratiosX96[i];
        expectedAmounts[i] = ratiosX96 == 0
```

```
            ? 0
            : FullMath.mulDiv(coefficientX96, ratiosX96, Q96);
        if (expectedAmounts[i] >= request.minAmounts[i]) continue;
        return (false, false, expectedAmounts);
    }
    for (uint256 i = 0; i < length; i++) {
        if (s.erc20Balances[i] >= expectedAmounts[i]) continue;
        return (true, false, expectedAmounts);
    }
    return (true, true, expectedAmounts);
}
```

In the calculation of `expectedAmount`, the `Mellow protocol`'s fee is also taken into account:

```
value = FullMath.mulDiv(value, D9 - s.feeD9, D9);
```

However, the lpAmountToAmount function looks like this:

```
function lpAmountToAmount(
    uint256 lpAmount,
    IMellowVault mellowVault
) public view returns (uint256) {
    if (lpAmount == 0) return 0;

    IMellowVault.ProcessWithdrawalsStack memory s = mellowVault
        .calculateStack();

    uint256 wstEthAmount = FullMath.mulDiv(
        FullMath.mulDiv(lpAmount, s.totalValue, s.totalSupply),
        s.ratiosX96[0],
        s.ratiosX96Value
    );
    return wstEthAmount;
}
```

It does not account for the `Mellow protocol`'s fee in the calculation, leading to incorrect token accounting in the `getTotalDelegated` calculations of contract `MellowHandler` and potentially causing an imbalance in the `vault`.

Recommendation

We recommend aligning the calculation of amount corresponding to `lpAmount` with the `Mellow protocol`.

Update

Client's response

Added Mellow fee subtraction as recommended

Oxorio's response

Fixed at `15824a9342d78b605934ee3c13de0329a504e9f9`.

C-02	Potential overflow in `tokenAmount` calculation in `IMellowRestaker`
Severity	**CRITICAL**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegate`	141

Description

In the `delegate` function of the `IMellowRestaker` contract, there is a potential for overflow when calculating `tokenAmount`. This could occur if the balance of `_asset` exceeds the `amount` parameter, leading to a transaction revert.

```
function delegate(
    uint256 amount,
    uint256 deadline
) external onlyTrustee whenNotPaused returns (uint256 tokenAmount, uint256 lpAmount) {
    ...
    uint256 returned = _asset.balanceOf(address(this));

    tokenAmount = amount - returned;
}
```

This issue can arise in the following scenarios:

- The balance of `_asset` is greater than amount at the time of the function call. This can happen if a withdrawal request from `Mellow protocol` has been processed but the `claimMellowWithdrawalCallback` has not been called.
- An attacker could perform a front-running attack on the `delegate` transaction by transferring an amount equal to `amount + 1`, making the `delegate` transaction fail.
- Tokens could have been mistakenly transferred to the contract's balance.

Recommendation

We recommend recalculating `tokenAmount` using the difference between the balance before and after the transaction (`balanceAfter - balanceBefore`). This approach will prevent overflow and ensure accurate calculations.

Update

Client's response

The delegate and delegateMellow method now finds the difference before and after as returned amount

Oxorio's response

Fixed at `15824a9342d78b605934ee3c13de0329a504e9f9`.

C-03	Returned tokens sent to `trusteeManager` address instead of `_vault` in `IMellowRestaker`
Severity	**CRITICAL**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegate`	142

Description

In the `delegate` function of the `IMellowRestaker` contract, returned tokens are sent to `msg.sender`, which could be either the `trusteeManager` or the `_vault`, as the function is restricted by the `onlyTrustee` modifier.

```
function delegateMellow(
    uint256 amount,
    uint256 deadline,
    address mellowVault
) external onlyTrustee whenNotPaused returns (uint256 lpAmount) {
    ...
    IERC20(_asset).safeTransfer(msg.sender, returned);
}
```

```
modifier onlyTrustee() {
    if (msg.sender != _vault && msg.sender != _trusteeManager)
        revert NotVaultOrTrusteeManager();
    _;
}
```

This behavior can lead to returned tokens being sent to the `trusteeManager` instead of the `Vault`, causing an imbalance in the `Vault`.

Recommendation

We recommend sending returned tokens directly to the `_vault` address instead of `msg.sender` to ensure proper token accounting and prevent vault imbalances.

Update

Client's response

Assets are now transferred explicitly to vault

Oxorio's response

Fixed at `15824a9342d78b605934ee3c13de0329a504e9f9`.

C-04	Potential `gas bomb` and blocking of `redeem` in `InceptionVault_S`
Severity	**CRITICAL**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `IMellowRestaker` > function `redeem`	264
InceptionVault_S.sol	contract `IMellowRestaker` > function `isAbleRedeem`	395

Description

In the `redeem` function of `InceptionVault_S`, there is a potential gas bomb due to the iterative search through `claimerWithdrawalsQueue`. As the queue grows, the number of iterations increases, leading to higher gas costs and potential transaction failures.

```
function redeem(address receiver) external whenNotPaused nonReentrant {
    (bool isAble, uint256[] memory availableWithdrawals) = isAbleToRedeem(
        receiver
    );
    if (!isAble) revert IsNotAbleToRedeem();

    // ...

function isAbleToRedeem(
    address claimer
) public view returns (bool able, uint256[] memory) {
    // get the general request
    uint256 index;
    Withdrawal memory genRequest = _claimerWithdrawals[claimer];

    uint256[] memory availableWithdrawals = new uint256[](
        epoch - genRequest.epoch
    );

    if (genRequest.amount == 0) return (false, availableWithdrawals);

    for (uint256 i = 0; i < epoch; ++i) {
```

```
        if (claimerWithdrawalsQueue[i].receiver == claimer) {
            able = true;

            availableWithdrawals[index] = i;
            ++index;
        }
    }
    // decrease arrays

    if (availableWithdrawals.length - index > 0)
        assembly {
            mstore(availableWithdrawals, index)
        }

    return (able, availableWithdrawals);
}
```

In the `isAbleToRedeem` function, the loop iterates over all entries in `claimerWithdrawalsQueue`, including those processed up to the current epoch. As the queue grows, the number of iterations increases, creating a gas bomb effect and potentially blocking withdrawals for end users.

Recommendation

We recommend optimizing the `redeem` function by tracking processed withdrawal requests for each user. This will reduce the number of iterations and prevent gas-related issues.

Update

Client's response

Added `_traversalEpoch` mapping variable to track last epoch index of traversal

Oxorio's response

The solution is implemented incorrectly.

1. The issue is that `_traversalEpoch` is only set after the call to `isAbleToRedeem`. This means that at the moment of the first call, the user is not protected from a gas bomb, and if this is a holder who has not interacted with the protocol for a long time, they will encounter problems.
2. At the start of the protocol, a revert is possible during `_traversalEpoch[receiver] = epoch - 1;` because `epoch` in this case is 0, and the `isAble` check is only called afterward.

To resolve the issue, the following options can be considered:

1. Store the numbers of unprocessed withdrawals in a separate variable. In `genRequest`, record the number of the last epoch in which the request was created.
2. Perform the check only for withdrawals that are less than or equal to the current epoch number.

Client's response

Revised Logic. Each user has now genRequest.withdrawals field. Incremented at withdraw and decremented at redeem. genRequest.epoch is also updated at withdraw to most recent epoch if `genRequest.withdrawals == 0`.

Oxorio's response

Partially fixed at `ca83f1a6e1e55e81ef6d542b64114632cfc894c0`.

The following comments remain:

- Optimizing the decrement of withdrawals[receiver] in the loop at #319
  Instead of decrementing withdrawals[receiver] by one in each iteration of the loop, you can reduce it by the number of processed withdrawals in a single operation. This will minimize the number of state changes and improve efficiency.

- Redundant condition `if (rEpoch < genRequest.epoch) revert EpochsMismatch();` at 441
  The condition `if (rEpoch < genRequest.epoch) revert EpochsMismatch();` seems redundant. In this case, it would be better to simply return early with empty values instead of reverting the transaction. This approach simplifies the logic and avoids unnecessary errors.

- Handling the case when `rEpoch > epoch` at 440
  The current logic if `(rEpoch < epoch) rEpoch++;` does not account for the situation where `rEpoch` is greater than the current epoch. In such cases, you should set `rEpoch = epoch` and process only the values available up to the current epoch. This ensures that the system works correctly even if `rEpoch` is ahead of the current epoch.

Oxorio's response

Fixed at `8d398d1f01a571a180aa17c805ee42ec5dfe158e`.

3.2 MAJOR

M-01	Missing return of undelegated tokens in `delegateMellow` function in `IMellowRestaker`
Severity	**MAJOR**
Status	• FIXED

Description

In the `delegateMellow` function, undelegated tokens are not returned to the Mellow protocol. This can lead to unexpected behavior, as unused tokens remain in the `IMellowRestaker` contract.

```
function delegateMellow(
    uint256 amount,
    uint256 deadline,
    address mellowVault
) external onlyTrustee whenNotPaused returns (uint256 lpAmount) {
    IMellowDepositWrapper wrapper = mellowDepositWrappers[mellowVault];
    if (address(wrapper) == address(0)) revert InactiveWrapper();
    _asset.safeTransferFrom(_vault, address(this), amount);

    IERC20(_asset).safeIncreaseAllowance(address(wrapper), amount);


    uint256 minAmount = (amount * (10000 - depositSlippage)) / 10000;
    return
        wrapper.deposit(
            address(this),
            address(_asset),
            amount,
            minAmount,
            block.timestamp + deadline
        );
}
```

The `deposit wrapper` has the following logic:

```
function deposit(
    address to,
    address token,
    uint256 amount,
    uint256 minLpAmount,
    uint256 deadline,
    uint256 referralCode
) external payable returns (uint256 lpAmount) {
  // ...
  uint256 balance = IERC20(wsteth).balanceOf(wrapper);
  if (balance > 0) IERC20(wsteth).safeTransfer(sender, balance);
}
```

This results in unused tokens remaining on the balance of the `IMellowRestaker` contract, leading to unexpected behavior.

Recommendation

We recommend returning undelegated tokens to the `Vault` to ensure proper token management and prevent unexpected behavior.

Update

Client's response

delegateMellow now returns undelegated tokens to vault

Oxorio's response

Fixed at `15824a9342d78b605934ee3c13de0329a504e9f9`.

M-02	Any user can call `claimCompletedWithdrawals` and `updateEpoch` in `MellowHandler`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
MellowHandler.sol	contract `MellowHandler` > function `claimCompletedWithdrawals`	103
MellowHandler.sol	contract `MellowHandler` > function `updateEpoch`	114

Description

In the mentioned locations, the functions `claimCompletedWithdrawals` and `updateEpoch` can be called by any user. This allows an attacker to perform a Denial of Service (DoS) attack on the contract by repeatedly calling these functions, potentially disrupting the normal operation of the protocol.

Recommendation

We recommend adding the `onlyOperator` modifier to restrict the execution of these functions to authorized operators only. This will prevent unauthorized users from calling these functions and mitigate the risk of DoS attacks.

Update

Client's response

Both functions are now restricted to onlyOperator

Oxorio's response

Fixed at `15824a9342d78b605934ee3c13de0329a504e9f9`.

M-03	Incorrect calculation of `minAmount` lp tokens in `IMellowRestaker`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegateMellow`	106
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegateMellow`	130

Description

In the mentioned locations, there is an incorrect calculation of `minAmount`. The `minAmount` is calculated as follows:

```
uint256 minAmount = (amount * (10000 - depositSlippage)) / 10000;
// ...
wrapper.deposit(
    address(this),
    address(_asset),
    amount,
    minAmount,
    block.timestamp + deadline
);
```

However, in the deposit function of the wrapper, `minAmount` represents the minimum amount of LP tokens expected after delegation, which depends on the rate in the `Mellow protocol`. This discrepancy can lead to incorrect calculations of the minimum LP tokens and may cause transaction failures.

Recommendation

We recommend pre-calculating the expected LP tokens using the `amountToLpAmount` function and then calculating `minAmount` based on this value. This will ensure accurate calculations and prevent transaction failures.

Update

Client's response

minAmount is now based on amountToLpAmount value

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

M-04	Inability to remove inactive mellow vault in `IMellowRestaker`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `changeMellowWrapper`	253

Description

In the `changeMellowWrapper` function of the `IMellowRestaker` contract, it is impossible to remove an inactive Mellow vault because the function does not allow setting `newDepositWrapper` to `address(0)`.

```
function changeMellowWrapper(address mellowVault, address newDepositWrapper) external
onlyOwner {
  if (mellowVault == address(0) || newDepositWrapper == address(0)) revert ZeroAddress();
  if (address(IMellowDepositWrapper(newDepositWrapper).vault()) != mellowVault) revert
InvalidWrapperForVault();

  address oldWrapper = address(mellowDepositWrappers[mellowVault]);
  if (oldWrapper == address(0)) revert NoWrapperExists();

  mellowDepositWrappers[mellowVault] = IMellowDepositWrapper(newDepositWrapper);

  emit WrapperChanged(mellowVault, oldWrapper, newDepositWrapper);
}
```

This restriction prevents the deactivation of a `Mellow vault` in cases where it has been compromised or is no longer operational.

Recommendation

We recommend allowing `newDepositWrapper` to be set to address(0) or adding a separate function to deactivate a vault. This will provide flexibility in managing inactive or compromised vaults.

Update

Client's response

Added deactivateMellow vault function to remove wrapper thus disabling delegation to deactivated vaults

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

Perhaps it is also worth considering a function that would allow manually withdrawing tokens from a compromised Vault or performing an emergency withdrawal from a compromised Vault.

Client's response

We had emergency withdrawal function that we deprecated because of mellow upgrade. However, we will consider emergency withdraw for mellow upgraded version of contracts.

M-05	Incorrect Calculation of `maxMint` in `InceptionVault_S`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `mint`	162

Description

In the mint function of `InceptionVault_S`, the calculation of `maxMint` includes a `depositBonus`, which leads to incorrect calculations when converting to `assetsAmount`.

```
 function mint(
    uint256 shares,
    address receiver
) external nonReentrant whenNotPaused returns (uint256) {
    uint256 maxShares = maxMint(msg.sender);
    if (shares > maxShares)
        revert ExceededMaxMint(receiver, shares, maxShares);

    uint256 assetsAmount = convertToAssets(shares);
    _deposit(assetsAmount, msg.sender, receiver);

    return assetsAmount;
}

function maxMint(address receiver) public view returns (uint256) {
    return
        !paused() ? previewDeposit(IERC20(asset()).balanceOf(receiver)) : 0;
}

function previewDeposit(uint256 assets) public view returns (uint256) {
    uint256 depositBonus;
    if (depositBonusAmount > 0) {
        depositBonus = calculateDepositBonus(assets);
        if (depositBonus > depositBonusAmount)
```

```
            depositBonus = depositBonusAmount;
        }

        return convertToShares(assets + depositBonus);
    }
```

The `maxMint` function includes the `depositBonus` in its calculation, allowing users to specify `iShares` amounts that include the bonus. The resulting `assetsAmount` is then passed to the `_deposit` function, which adds the `depositBonus` again:

```
function _deposit(
    uint256 amount,
    address sender,
    address receiver
) internal returns (uint256) {
    // ...
    uint256 iShares = convertToShares(amount + depositBonus);
    inceptionToken.mint(receiver, iShares);
}
```

This double-counting of the `depositBonus` causes the `maxMint` limitation to function incorrectly.

Recommendation

We recommend revising the logic for calculating `maxMint` to exclude the `depositBonus` and ensure accurate conversions.

Update

Client's response

Added 4th parameter for _deposit(), "calculate". A false flag is passed when mint calls _deposit() and thus depositBonus is not recalculated and added again to assetsAmount which already includes the bonus in mint.

Oxorio's response

This does not solve the problem. Ultimately, you are not adding any bonus except in the situation where shares are transferred as shares = maxShares. In all other cases, you are now not accruing any bonus. In this situation, it is worth considering revising the logic of the maxMint function specifically.

Client's response

`maxMint` now does not include `depositBonus` as suggested.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

M-06	Potential enial of Service (DoS) in `InceptionVault_S`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `withdraw`	225
InceptionVault_S.sol	contract `InceptionVault_S` > function `setWithdrawMinAmount`	574

Description

In `InceptionVault_S`, setting `withdrawMinAmount` to `0` can allow an attacker to fill the `claimerWithdrawalsQueue` with zero values. This can lead to a potential Denial of Service (DoS) attack, as the queue may become bloated with invalid or zero-value entries, causing inefficiencies or failures in processing legitimate withdrawal requests.

Recommendation

We recommend adding a validation check in the `setWithdrawMinAmount` function to ensure that the value is non-zero. This will prevent the queue from being filled with zero values and mitigate the risk of a DoS attack.

Update

Client's response

Added validation check for 0 value.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

M-07	Inability to remove inactive vault in `ISymbioticRestaker`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	-	-

Description

In the `ISymbioticRestaker` contract, it is impossible to remove an inactive Mellow vault. This functionality prevents the deactivation of a Mellow vault in cases where it has been compromised or is no longer operational.

Recommendation

We recommend adding a separate function to deactivate a vault. This will provide flexibility in managing inactive or compromised vaults.

Update

Client's response

Added function to remove vaults

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

M-08	`trusteeManager` cannot execute transaction in `ISymbioticRestaker`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker`	79
IMellowRestaker.sol	contract `IMellowRestaker`	101

Description

In the mentioned locations, the `trusteeManager` cannot make a call because there is no `approve` from the `_vault` address, which is a contract during the `transferFrom` operation.

```
_asset.safeTransferFrom(_vault, address(this), amount);
```

Recommendation

We recommend modifying this function so that the `trusteeManager` can execute the transaction correctly.

Update

Client's response

Replaced _vault with `msg.sender`.

Oxorio's response

There appears to be a critical inconsistency in the flow of assets when `trusteeManager` is involved. The current implementation suggests that when `trusteeManager` provides assets for staking in Symbiotic, these assets bypass the standard Vault flow. This creates a potential conflict with C-03 specification, as the returned balance should logically be directed to the original provider of the assets (`msg.sender`) rather than the Vault.
It is necessary to examine the functionality of `trusteeManager` to determine whether we

correctly understand that trusteeManager provides its funds for staking in Symbiotic, bypassing the Vault. If this is the case, it requires further analysis.

Oxorio's response

Fixed at `bfae89718774a612ba713f23519a984389eeefaa`.

M-09	Missing setup of `symbioticRestaker` in `SymbioticHandler`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
SymbioticHandler.sol	contract `SymbioticHandler` > function `__SymbioticHandler_init`	53

Description

In the `__SymbioticHandler_init` function of the `SymbioticHandler` contract, the `_mellowRestaker` contract is set up, but the `_symbioticRestaker` is not. This will cause some functions and systems to fail, such as `getTotalDeposited` and `getTotalDelegated`, as well as contracts dependent on these functions.

Recommendation

We recommend adding the setup for `_symbioticRestaker`.

Update

Client's response

`SymbioticHandler` now initializes `_symbioticRestaker`.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

M-10	Missing check that `amount` does not exceed `freeBalance` in `InceptionVault_S`
Severity	**MAJOR**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `delegateToSymbioticVault`	193

Description

In the `delegateToSymbioticVault` function of the `InceptionVault_S` contract, the `_beforeDeposit` function is commented out. This function checks that the `amount` does not exceed the `freeBalance`.

```
    function _beforeDeposit(uint256 amount) internal view {
        uint256 freeBalance = getFreeBalance();
        if (amount > freeBalance) revert InsufficientCapacity(freeBalance);
    }
```

This could lead to system imbalance and unexpected consequences.

Recommendation

We recommend adding the `_beforeDeposit` check.

Update

Client's response

Uncommented the check

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

3.3 WARNING

W-01	Potential overflows and division by zero in `InceptionLibrary`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
InceptionLibrary.sol	contract `InceptionLibrary` > function `calculateDepositBonus`	17
InceptionLibrary.sol	contract `InceptionLibrary` > function `calculateWithdrawalFee`	51

Description

In the mentioned locations, there are potential overflows and division-by-zero risks, which could lead to transaction reverts. This is possible because:

- The values `maxDepositBonusRate` and `optimalBonusRate`
- The values `maxFlashWithdrawalFeeRate` and `optimalFeeRate` are not validated against each other.
- Subtraction operations are not checked for underflow.

```
uint256 bonusPercent = maxDepositBonusRate -
                (bonusSlope * (capacity + replenished / 2)) /
                targetCapacity;
...

uint256 bonusPercent = maxFlashWithdrawalFeeRate -
                (feeSlope * (capacity - amount / 2)) /
                targetCapacity;
```

- `targetCapacity` is not checked for zero values.

Recommendation

We recommend adding value checks to prevent potential overflows and division-by-zero errors.

Update

Client's response

Added validation checks for overflows and division-by-zero as requires.

Oxorio's response

The following checks are missing:

- In this line, `capacity` and `amount` are not validated.
- In this line, there is no check to ensure that `optimalCapacity != 0`.

Client's response

Added both mentioned checks.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-02	Missing validation of set values in `InceptionVault_S`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionLibrary` > function `setDepositBonusParams`	499
InceptionVault_S.sol	contract `InceptionLibrary` > function `setFlashWithdrawFeeParams`	522

Description

In the mentioned locations, there is no validation of values relative to each other.

```
function setDepositBonusParams(
    uint64 newMaxBonusRate,
    uint64 newOptimalBonusRate,
    uint64 newDepositUtilizationKink
) external onlyOwner {
    if (newMaxBonusRate > MAX_PERCENT)
        revert ParameterExceedsLimits(newMaxBonusRate);
    if (newOptimalBonusRate > MAX_PERCENT)
        revert ParameterExceedsLimits(newOptimalBonusRate);
    if (newDepositUtilizationKink > MAX_PERCENT)
        revert ParameterExceedsLimits(newDepositUtilizationKink);

    maxBonusRate = newMaxBonusRate;
    optimalBonusRate = newOptimalBonusRate;
    depositUtilizationKink = newDepositUtilizationKink;

    emit DepositBonusParamsChanged(
        newMaxBonusRate,
        newOptimalBonusRate,
        newDepositUtilizationKink
    );
}

function setFlashWithdrawFeeParams(
    uint64 newMaxFlashFeeRate,
```

```
        uint64 newOptimalWithdrawalRate,
        uint64 newWithdrawUtilizationKink
    ) external onlyOwner {

        if (newMaxFlashFeeRate > MAX_PERCENT)
            revert ParameterExceedsLimits(newMaxFlashFeeRate);
        if (newOptimalWithdrawalRate > MAX_PERCENT)
            revert ParameterExceedsLimits(newOptimalWithdrawalRate);
        if (newWithdrawUtilizationKink > MAX_PERCENT)
            revert ParameterExceedsLimits(newWithdrawUtilizationKink);

        maxFlashFeeRate = newMaxFlashFeeRate;
        optimalWithdrawalRate = newOptimalWithdrawalRate;
        withdrawUtilizationKink = newWithdrawUtilizationKink;

        emit WithdrawFeeParamsChanged(
            newMaxFlashFeeRate,
            newOptimalWithdrawalRate,
            newWithdrawUtilizationKink
        );
    }
```

- The parameters `maxFlashFeeRate` and `optimalWithdrawalRate` are passed to the `InceptionLibrary` and could cause overflows.
- The parameters `maxDepositFeeRate` and `optimalDepositRate` are also passed to the `InceptionLibrary` and could cause overflows.

Recommendation

We recommend adding checks to ensure that values are validated relative to each other to avoid potential issues.

Update

Client's response

Added validation checks for overflows and division-by-zero as requires.

Oxorio's response

Fixed at `486b82dbcccbee50b13b226547eed56f75ffcbeb`.

W-03	Missing validation of `ratio` in `InceptionVault_S`
Severity	**WARNING**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionLibrary` > function `ratio`	411

Description

In `InceptionVault_S`, there is no validation for the ratio value:

```
function ratio() public view returns (uint256) {
    return ratioFeed.getRatioFor(address(inceptionToken));
}
```

If the `ratioFeed` is compromised or updated to an incorrect value, the `ratio` could become excessively large or small, leading to potential attacks.

Recommendation

We recommend adding a maximum deviation check from the previous value when retrieving the `ratio`.

Update

Client's response

The recommendation has been read through and we think real time ratio access is important to reflect accurate yield. And that we are optimistic on the ratio source.

W-04	Unnecessary 1 wei addition in fee calculation in `InceptionLibrary`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
InceptionLibrary.sol	contract `InceptionLibrary` > function `calculateWithdrawalFee`	79

Description

In the `calculateWithdrawalFee` function of the `InceptionLibrary`, there is an unnecessary addition of 1 wei:

```
function calculateWithdrawalFee(
    uint256 amount,
    uint256 capacity,
    uint256 optimalCapacity,
    uint256 optimaFeeRate,
    uint256 maxFlashWithdrawalFeeRate,
    uint256 targetCapacity
) external pure returns (uint256 fee) {

    /// @dev the utilization rate is in the range [100:25] %
    if (amount > 0 && capacity > optimalCapacity) {
        uint256 replenished = amount;
        if (capacity - amount < optimalCapacity)
            replenished = capacity - optimalCapacity;

        fee += (replenished * optimaFeeRate) / MAX_PERCENT;
        amount -= replenished;
        capacity -= replenished;
        if (fee == 0) ++fee;
    }
    /// @dev the utilization rate is in the range [25:0] %
    if (amount > 0) {
        uint256 feeSlope = ((maxFlashWithdrawalFeeRate - optimaFeeRate) *
            1e18) / ((optimalCapacity * 1e18) / targetCapacity);
```

```
            uint256 bonusPercent = maxFlashWithdrawalFeeRate -
                (feeSlope * (capacity - amount / 2)) /
                targetCapacity;
            fee += (amount * bonusPercent) / MAX_PERCENT;
            if (fee == 0) ++fee;
        }
        if (fee == 0) ++fee;
    }
```

This leads to a situation where, if the calculated fee is 0, it is set to 1 wei. In the redeem function of `InceptionVault_S`, the condition:

```
if (fee == 0) revert ZeroFlashWithdrawFee();
uint256 protocolWithdrawalFee = (fee * protocolFee) / MAX_PERCENT;
```

is bypassed, and `protocolWithdrawalFee` could end up being 0, leading to unnecessary computations.

Recommendation

We recommend removing the unnecessary addition of 1 wei.

Update

Client's response

The recommendation has been read through and +1 wei is added for rounding and dust related issues.

Oxorio's response

Could you provide a detailed explanation regarding dust?

Client's response

The addition of wei is because in some cases users can withdraw small amounts or dust amounts without fees if it returned 0. However, the check at `_flashWithdraw` function is redundant and has been removed.

Oxorio's response

Fixed at `e11da47d070c2bd9482af19644ce415679bf73d9`.

W-05	Missing maximum value of `newTargetCapacity` check in `MellowHandler`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
MellowHandler.sol	contract `MellowHandler` > function `setTargetFlashCapacity`	215

Description

In the `setTargetFlashCapacity` function of the `MellowHandler` contract, there is no check to ensure that the value `newTargetCapacity` does not exceed `MAX_TARGET_PERCENT`.

Recommendation

We recommend adding a check to ensure that the value `newTargetCapacity` does not exceed `MAX_TARGET_PERCENT`.

Update

Client's response

Added validation check to ensure the value newTargetCapacity does not exceeed MAX_TARGET_PERCENT

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`

W-06	Incorrect value passed to Custom Error `Insufficient Capacity` in `MellowHandler`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
MellowHandler.sol	contract `MellowHandler` > function `_beforeDeposit`	61

Description

In the `_beforeDeposit` function of the `MellowHandler` contract, the custom error `InsufficientCapacity` is passed an incorrect value:

```
function _beforeDeposit(uint256 amount) internal view {
    if (amount > getFreeBalance())
        revert InsufficientCapacity(totalAssets());
}
```

The error is triggered when amount exceeds `getFreeBalance`, but `totalAssets()` is passed instead of `getFreeBalance`.

Recommendation

We recommend passing the correct value, `getFreeBalance`, to the error.

Update

Client's response

Replaced with getFreeBalance()

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3`.

W-07	Missing check for active or existing vault in `IMellowRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `changeAllocation`	253

Description

In the `changeAllocation` function of the `IMellowRestaker` contract, there is no check to ensure that the `vault` is active or exists. This could lead to unexpected behavior when setting allocations for a vault that has not yet been configured.

Recommendation

We recommend adding a check to ensure that the `vault` is active or exists before setting allocations.

Update

Client's response

Added validation check for invalid vault address

Oxorio's response

The solution appears inefficient and incorrect. In the `deactivateMellowVault` function, deactivation is performed by setting the `depositWrapper` value to `address(0)`, while the vault existence check does not verify the `depositWrapper`.
We recommend calling the function only for those vaults that are activated and exist, meaning that the wrapper exists and the vault is present in the list.

Client's response

Added wrapper existence check.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`. But this loop looks redundant because you can check if the `depositWrapper` exists for `mellowVault`.

Oxorio's response

Fixed at `8d398d1f01a571a180aa17c805ee42ec5dfe158e`.

W-08	Missing check for zero amount and `vault` existence in `IMellowRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `amountToLpAmount`	307

Description

In the `amountToLpAmount` function of the `IMellowRestaker` contract, there is no check to ensure that the amount is not zero or that the vault exists. This could lead to unnecessary computations.

Recommendation

We recommend adding checks to ensure that the amount is not zero and that the vault exists.

Update

Client's response

Partially fixed. Checks the amount only. For the vault address, we rely on internal argument passing until mellow upgrades to MellowV2 because, these conversions from amountToLp and vice versa are very gas inefficient As soon as, they will migrate to V2 and give us ratio feed, we will upgrade this check.

Oxorio's response

Partially fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`. As a temporary solution, you can check your mapping to ensure that a wrapper exists, since this is added on behalf of the owner. This way, external integration with you can avoid potential issues.

Client's response

Added vault's existence check

Oxorio's response

Fixe at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-09	Unsafe functions in `InceptionToken`
Severity	**WARNING**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionToken.sol	contract `InceptionToken`	-

Description

`OpenZeppelin` library is being used in the contract `InceptionToken`, but the `OpenZeppelin` version is not specified. Based on the Solidity version 0.8.4 being used, we assume it's `OpenZeppelin` version 4.9.
This version contains two potentially unsafe functions: `increaseAllowance` and `decreaseAllowance`, which are not part of the ERC20 standard and were later removed. The discussion about this can be found here: openzeppelin-contracts/issues/4583.

Recommendation

We recommend disabling these functions.

Update

Client's response

The recommendation has been read through and it seems the security concerns that fix increaseAllowance and decreaseAllowance are not critical nor high in the wild.

W-10	Deployer becomes the default owner of the contract in `InceptionToken`
Severity	**WARNING**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionToken.sol	contract `InceptionToken` > function `initialize`	50

Description

In the `initialize` function of the `InceptionToken` contract, the owner is set by default to the `msg.sender` address, i.e., the deployer.

Recommendation

We recommend adding a separate parameter to set the owner's address, as well as importing the Ownable2Step contract, as done in the `InceptionVault_S` contract.

Update

Client's response

The recommendation has been read through and do ownership transfer after deployment. `Ownable2Step` is nice to have but we think `Ownable` is sufficient.

W-11	Treasury address set to deployer in `InceptionVault_S`
Severity	**WARNING**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionToken` > function `__InceptionVault_init`	86

Description

In function `__InceptionVault_init` of contract `InceptionVault_S`, the treasury address is set to `msg.sender`, which is the deployer.

Recommendation

We recommend passing the treasury address as a separate parameter distinct from the deployer.

Update

Client's response

The recommendation has been read through and we do set treasury address after deployment.

W-12	Missing check for vault existence in `_vaults` list in `ISymbioticRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `getDeposited`	139

Description

In the `getDeposited` function of the `ISymbioticRestaker` contract, there is no check to ensure that `_vault` exists in the `_vaults` list.

Recommendation

We recommend adding a check to ensure that `_vault` is in the `_vaults` list.

Update

Client's response

Added vault's existence check.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-13	Missing `collateral` check for vault in `ISymbioticRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `initialize`	68
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `addVault`	167

Description

In the mentioned locations, when adding a vault, there is no check to ensure that the vault's `collateral` matches the `_asset` of the contract. This could lead to incorrect system behavior.

Recommendation

We recommend adding a check to ensure that the `collateral` of the symbiotic vault matches `_asset`.

Update

Client's response

Added collateral check

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-14	Missing duplicate check in `_vaults` array in `ISymbioticRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `initialize`	68

Description

In the `initialize` function of the `ISymbioticRestaker` contract, there is no check to ensure that the vault is not already in the `_vaults` array.

Recommendation

We recommend adding a duplicate check, as is done in the `addVault` function.

Update

Client's response

Added duplicate check

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-15	Missing setup of `_vault` in `ISymbioticRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `initialize`	57
IMellowRestaker.sol	contract `IMellowRestaker` > function `initialize`	65

Description

In the mentioned contracts, the `initialize` function does not set the value of `_vault`.

Recommendation

We recommend adding the setup of `_vault` in the `initialize` function.

Update

Client's response

Adding setup of `_vault`.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

W-16	Incorrect `amount` value in `SymbioticHandler`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
SymbioticHandler.sol	contract `SymbioticHandler` > function `undelegateFromSymbiotic`	123

Description

In the `undelegateFromSymbiotic` function of the `SymbioticHandler` contract, an incorrect `amount` value is used.

```
        amount = symbioticRestaker.withdraw(vault, amount);
        emit StartMellowWithdrawal(address(symbioticRestaker), amount);
```

The `amount` is passed through the `withdraw` function of the `ISymbioticRestaker` contract,

```
    function withdraw(address vaultAddress, uint256 amount)
        external
        onlyTrustee
        whenNotPaused
        returns (uint256)
    {
        require(_vaults.contains(vaultAddress), InvalidVault());
        require(withdrawals[vaultAddress] == 0, WithdrawalInProgress());

        IVault vault = IVault(vaultAddress);
        (, uint256 mintedShares) = vault.withdraw(address(this), amount);
        withdrawals[vaultAddress] = vault.currentEpoch() + 1;
        return mintedShares;
    }
```

which returns the `mintedShares` value obtained after calling the `withdraw` function of the `SymbioticVault`. However, this function returns `mintedShares` relative to the total

amount in the epoch, according to the documentation, and this number does not correspond to the `amount`.

Recommendation

We recommend refactoring the code to return the correct values.

Update

Client's response

The input parameter amount has been used as a return value because symbiotic subtracts exact `amount` as `withdrawnAssets` from corresponding variables in their contract. The amount can be achieved by conversion however, this seems unnecessary.

Oxorio's response

Fixed at `cf7ab740aebbed70a828f4a76bd763ef46bd0a25`.

W-17	Incorrect epoch handling condition in `ISymbioticRestaker`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `withdraw`	96

Description

In the `withdraw` function of the `ISymbioticRestaker` contract, there is a condition:

```
require(withdrawals[vaultAddress] == 0, WithdrawalInProgress());
withdrawals[vaultAddress] = vault.currentEpoch() + 1;
```

This creates a situation where all new `withdraw` calls cannot be executed, and the future epoch will not be processed.
In Symbiotic, an epoch lasts on average 7 days. With the current logic, users who initiate a withdrawal after the operator sends a withdrawal request to Symbiotic will have to wait an average of 2 weeks.

Recommendation

We recommend considering epoch handling so that if `withdrawals[vaultAddress] == vault.currentEpoch() + 1`, the requests in Symbiotic are summed up. Otherwise, revert until the requests for the current epoch are claimed.

Update

Client's response

The condition has been changed. If in the same epoch, many withdrawals are required, it simply allows it unless the epoch has elapsed in which case claim is mandatory before withdrawing in the updated epoch.

Oxorio's response

Fixed at `cf7ab740aebbed70a828f4a76bd763ef46bd0a25`.

W-18	Incorrect event in `SymbioticHandler`
Severity	**WARNING**
Status	• FIXED

Location

File	Location	Line
SymbioticHandler.sol	contract `SymbioticHandler` > function `undelegateFromSymbiotic`	123

Description

In the function `undelegateFromSymbiotic` of the `SymbioticHandler` contract, the event `StartMellowWithdrawal` is mistakenly emitted, even though the action relates to a `Symbiotic Vault`. This is semantically incorrect, as the behavior being represented should be tied to a corresponding `Symbiotic`-specific event.

Recommendation

Replace the `StartMellowWithdrawal` event with a more appropriate event, such as `StartSymbioticWithdrawal`, or create a new event specifically for `Symbiotic Vault` withdrawal interactions. Ensure the event accurately reflects the logic executed within the function, including the relevant parameters relating to the vault and delegation process.

Update

Client's response

Added new event `StartSymbioticWithdrawal`.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

3.4 INFO

I-01	`ReentrancyGuard` initialized but not used in `IMellowRestaker`
Severity	**INFO**
Status	• ACKNOWLEDGED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `initialize`	73

Description

In the `initialize` function of the `IMellowRestaker` contract, the `ReentrancyGuard` is initialized, but the `nonReentrant` modifier is not used anywhere.

Recommendation

We recommend adding the `nonReentrant` modifier where necessary or removing the import and initialization of `ReentrancyGuard`.

Update

Client's response

The recommendation has been read through and we are aware of this. To avoid storage slots being messed up, we kept the inheritance.

I-02	Incorrect condition in `MellowHandler`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
MellowHandler.sol	contract `MellowHandler` > function `setTargetFlashCapacity`	206

Description

In the `setTargetFlashCapacity` function of the `MellowHandler` contract, the condition:

```
if (newTargetCapacity <= 0) revert InvalidTargetFlashCapacity();
```

is incorrect because `newTargetCapacity` is of type `uint256` and cannot be less than `0`.

Recommendation

We recommend changing the condition to check if `newTargetCapacity == 0`.

Update

Client's response

Changed to equality comparison rather than relational

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-03	Missing check for amount not equal to 0 in `MellowHandler`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
MellowHandler.sol	contract `MellowHandler` > function `_depositAssetIntoMellow`	64
MellowHandler.sol	contract `MellowHandler` > function `undelegateFrom`	79

Description

In the mentioned locations of the `MellowHandler` contract, there is no check to ensure that the amount parameter is not equal to `0`.

Recommendation

We recommend adding a check to ensure that the amount is not equal to `0`.

Update

Client's response

For _depositAssetIntoMellow, no need to check for zero amount since it is already being checked in InceptionVault_S.delegateToMellowVault

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-04	Incorrect order of operations in `IMellowRestaker`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `withdrawMellow`	156

Description

In the function `withdrawMellow` of contract `IMellowRestaker`

```
IMellowVault mellowVault = IMellowVault(_mellowVault);
uint256 lpAmount = amountToLpAmount(amount, mellowVault);
uint256[] memory minAmounts = new uint256[](1);

minAmounts[0] = (amount * (10000 - withdrawSlippage)) / 10000; // slippage

if (address(mellowDepositWrappers[_mellowVault]) == address(0))
    revert InvalidVault();
```

```
if (address(mellowDepositWrappers[_mellowVault]) == address(0))
    revert InvalidVault();
```

The check should be performed before calculating `amountToLpAmount` to avoid unnecessary gas consumption.

Recommendation

We recommend reordering the operations.

Update

Client's response

Check is now before amountToLpAmount

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-05	Redunant in `InceptionToken`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
InceptionToken.sol	-	9

Description

In `InceptionToken`, there is a redundant import of `Convert`.

Recommendation

We recommend removing the redundant import.

Update

Client's response

Removed.

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-06	Treasury cannot be zero in `InceptionVault_S`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `_beforeWithdraw`	207

Description

In `InceptionVault_S`

```
if (treasury == address(0)) revert InceptionOnPause();
```

is redundant because treasury cannot be `address(0)`. During contract initialization, treasury is set to `msg.sender`, and the `setTreasuryAddress` function includes a check to ensure the address is not `address(0)`.

Recommendation

We recommend removing the redundant condition.

Update

Client's response

Removed.

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3`.

I-07	`redeem` and `flashWithdraw` have identical functionality in `InceptionVault_S`
Severity	**INFO**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `redeem`	246
InceptionVault_S.sol	contract `InceptionVault_S` > function `flashWithdraw`	305

Description

In `InceptionVault_S`, the redeem and flashWithdraw functions have identical functionality but emit different events.

```
function redeem(
    uint256 shares,
    address receiver,
    address owner
) external nonReentrant whenNotPaused returns (uint256 assets) {

    if (owner != msg.sender) revert MsgSenderIsNotOwner();
    __beforeWithdraw(receiver, shares);
    assets = convertToAssets(shares);
    uint256 fee;


    (assets, fee) = _flashWithdraw(shares, receiver, owner);

    emit Withdraw(owner, receiver, owner, assets, shares);
    emit WithdrawalFee(fee);

    return assets;
}

function flashWithdraw(
    uint256 iShares,
    address receiver
```

```
) external whenNotPaused nonReentrant {
    __beforeWithdraw(receiver, iShares);
    address claimer = msg.sender;
    (uint256 amount, uint256 fee) = _flashWithdraw(
        iShares,
        receiver,
        claimer
    );
    emit FlashWithdraw(claimer, receiver, claimer, amount, iShares, fee);
}
```

Recommendation

We recommend removing the redundant function to avoid duplication or extracting common logic into a separate internal function.

Update

Client's response

The recommendation has been read through and we are aware of this. The original function was flashWithdraw but to make the contract compatible with ERC4626, we had to have two functions with different signatures and events.

Oxorio's response

If your goal is full compatibility with the ERC4626 interfaces, you are missing the following functions:

- `previewMint`
- `maxWithdraw`
- `previewWithdraw`

and ERC20 specific functions:

- `totalSupply`
- `balanceOf`
- `decimals`
- `symbol`

Client's response

We are aiming compatibility with ERC4626 as much as possible. However, there are some scenarios like unpredictable withdraw amount and limitations of MellowV1 and some gas related issues, we couldn't go full towards full compatibility. We are on our way to full compatibility since Mellow has upgraded their contracts.

I-08	`protocolWithdrawalFee` can be zero in `InceptionVault_S`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `flashWithdraw`	339

Description

In function of contract `InceptionVault_S`, `protocolWithdrawalFee` can be zero. This may lead to an empty transfer to the treasury address.

Recommendation

We recommend adding a check to ensure `protocolWithdrawalFee` is not zero before transferring to the treasury.

Update

Client's response

Added validation check against zero value

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-09	Missing check for zero values in `InceptionVault_S`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `setWithdrawMinAmount`	574
InceptionVault_S.sol	contract `InceptionVault_S` > function `setDepositMinAmount`	579
InceptionVault_S.sol	contract `InceptionVault_S` > function `setFlashMinAmont`	584

Description

In the mentioned locations, there is no validation to ensure that the passed values are not zero. This could lead to unpredictable behavior and potential DDoS attacks.

Recommendation

We recommend adding validation to ensure that the passed values are not zero.

Update

Client's response

Added validation checks.

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-10	Typo in `InceptionVault_S`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
InceptionVault_S.sol	contract `InceptionVault_S` > function `setFlashMinAmont`	584

Description

In the function `setFlashMinAmont` of contract `InceptionVault_S` there is a typo in name.

Recommendation

We recommend changing name of function to `setFlashMinAmount`.

Update

Client's response

Fixed typo.

Oxorio's response

Fixed at `a1269302d485cb3185b63cbe6b086b78c943aa3e`.

I-11	Missing parameter validation in `InceptionVault_S`, `IMellowRestaker`
Severity	**INFO**
Status	• ACKNOWLEDGED

Location

File	Location	Line
InceptionVault_S.sol	__InceptionVault_init	56
IMellowRestaker.sol	contract `IMellowRestaker` > function `initialize`	66

Description

In the mentioned locations, there is no validation to ensure that the passed addresses are not `address(0)`.

Recommendation

We recommend adding validation to ensure that the passed addresses are not `address(0)`.

Update

Client's response

The recommendation has been read through and we are aware of this. We make sure no 0 address is passed in our deployment process.

I-12	Missing check that `sEpoch` has been processed or `sEpoch` has not yet occurred in `ISymbioticRestaker`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `claim`	115

Description

In the `claim` function of the `ISymbioticRestaker` contract, there is no check to ensure that the withdrawal has already been claimed in the epoch or that the epoch has not yet occurred. This leads to unnecessary gas costs.

Recommendation

We recommend adding a check to ensure that `sEpoch` has been processed or the epoch has not yet occurred.

Update

Client's response

Added checks for sEpoch

Oxorio's response

Fixed at `cf7ab740aebbed70a828f4a76bd763ef46bd0a25`.

I-13	`IBaseRestaker` Is not used
Severity	**INFO**
Status	• ACKNOWLEDGED

Location

File	Location	Line
IBaseRestaker.sol3	-	-

Description

The `IBaseRestaker` contract is not used in the codebase, even though it contains common implementations for both `IMellowRestaker` and `ISymbioticRestaker`.

Recommendation

We recommend using `IBaseRestaker` in the `IMellowRestaker` and `ISymbioticRestaker` contracts.

Update

Client's response

The IBaseRestaker is a new interface. We are going to make it standard base restaker contract in our upcoming upgrade

I-14	`_asset` Is already `IERC20` in `IMellowRestaker` and `InceptionVault_S`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegateMellow`	108
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegateMellow`	120
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegate`	141
IMellowRestaker.sol	contract `IMellowRestaker` > function `delegate`	158
ISymbioticRestaker.sol	contract `ISymbioticRestaker.sol` > function `delegateMellow`	85
InceptionVault_S.sol	contract `InceptionVault_S` > function `maxDeposit`	476
InceptionVault_S.sol	contract `InceptionVault_S` > function `maxMint`	482

Description

In the mentioned locations, the `IERC20(_asset)` construction is used, even though `_asset` is already an `IERC20` variable, not an address.

Recommendation

We recommend replacing `IERC20(_asset)` with `_asset`.

Update

Client's response

Removed all redundant type casting.

Oxorio's response

Fixed at `ca83f1a6e1e55e81ef6d542b64114632cfc894c0`.

I-15	Redundant Interface in `ISymbioticRestaker.sol`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	-	14

Description

In `ISymbioticRestaker.sol#L14`, the `IStakerRewards` interface is redundant and unused.

Recommendation

We recommend removing the redundant interface.

Update

Client's response

Removed.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

I-16	Incorrect variable naming in `ISymbioticRestaker`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker`	36

Description

In the `ISymbioticRestaker` contract, there are two variables: `_vaults` and `_vault`. They serve different purposes: `_vaults` refers to `symbiotic vaults`, while `_vault` is associated with `InceptionVault_S`.

Recommendation

We recommend renaming `_vaults` to `symbioticVaults`, following the convention used in `IMellowRestaker`.

Update

Client's response

Renamed to `_symbioticVaults`.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

I-17	Use custom errors instead of `require` in `ISymbioticRestaker`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > modifier `onlyTrustee`	45
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `delegate`	95
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `withdraw`	95
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `withdraw`	96
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `claim`	111
ISymbioticRestaker.sol	contract `ISymbioticRestaker` > function `claim`	112

Description

The code at these locations uses `require` statements, whereas the rest of the project utilizes custom errors for validation and error handling.

Recommendation

We recommend replacing `require` with custom errors to maintain consistency throughout the codebase and to better handle offchain error scenarios.

Update

Client's response

Replaced all require with if.

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c0`.

I-18	Incorrect comment in `SymbioticHandler`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
SymbioticHandler.sol	contract `SymbioticHandler`	112

Description

The comments at this locations incorrectly refer to `Mellow` instead of `Symbiotic`.

Recommendation

We recommend updating the comments to correctly reference `Symbiotic`.

Update

Client's response

Corrected

Oxorio's response

Fixed at `6f7c6cc9e7a102722e8280d806a08117982f11c`.

I-19	`TODO` comments in `ISymbioticRestaker`, `InceptionVault_S`, `SymbioticHandler`
Severity	**INFO**
Status	• ACKNOWLEDGED

Location

File	Location	Line
ISymbioticRestaker.sol	contract `ISymbioticRestaker`	131
SymbioticHandler.sol	contract `SymbioticHandler`	45
SymbioticHandler.sol	contract `SymbioticHandler` > function `undelegateFromSymbiotic`	125
InceptionBasicStrategyVault.sol	-	72
InceptionVault_S.sol	contract `InceptionVault_S` > function `delegateToSymbioticVault`	193

Description

`TODO` comments are present in the mentioned locations within the codebase.

Recommendation

We recommend removing the `TODO` comments and implementing the missing functionality to ensure the completeness of the codebase.

Update

Client's response

All these comments will be removed and code base will be refactored in the next iteration.

4. APPENDIX

4.1 SECURITY ASSESSMENT METHODOLOGY

Oxorio's smart contract security audit methodology is designed to ensure the security, reliability, and compliance of smart contracts throughout their development lifecycle. Our process integrates the Smart Contract Security Verification Standard (SCSVS) with our advanced techniques to address complex security challenges. For a detailed look at our approach, please refer to the full version of our methodology. Here is a concise overview of our auditing process:

1. Project Architecture Review

All necessary information about the smart contract is gathered, including its intended functionality and dependencies. This stage sets the foundation by reviewing documentation, business logic, and initial code analysis.

2. Vulnerability Assessment

This phase involves a deep dive into the smart contract's code to identify security vulnerabilities. Rigorous testing and review processes are applied to ensure robustness against potential attacks.

This stage is focused on identifying specific vulnerabilities within the smart contract code. It involves scanning and testing the code for known security weaknesses and patterns that could potentially be exploited by malicious actors.

3. Security Model Evaluation

The smart contract’s architecture is assessed to ensure it aligns with security best practices and does not introduce potential vulnerabilities. This includes reviewing how the contract integrates with external systems, its compliance with security best practices, and whether the overall design supports a secure operational environment.

This phase involves a analysis of the project's documentation, the consistency of business logic as documented versus implemented in the code, and any assumptions made during the design and development phases. It assesses if the contract's architectural design adequately addresses potential threats and integrates necessary security controls.

4. Cross-Verification by Multiple Auditors

Typically, the project is assessed by multiple auditors to ensure a diverse range of insights and thorough coverage. Findings from individual auditors are cross-checked to verify accuracy and completeness.

5. Report Consolidation

Findings from all auditors are consolidated into a single, comprehensive audit report. This report outlines potential vulnerabilities, areas for improvement, and an overall assessment of the smart contract's security posture.

6. Reaudit of Revised Submissions

Post-review modifications made by the client are reassessed to ensure that all previously identified issues have been adequately addressed. This stage helps validate the effectiveness of the fixes applied.

7. Final Audit Report Publication

The final version of the audit report is delivered to the client and published on Oxorio's official website. This report includes detailed findings, recommendations for improvement, and an executive summary of the smart contract's security status.

4.2 CODEBASE QUALITY ASSESSMENT REFERENCE

The tables below describe the codebase quality assessment categories and rating criteria used in this report.

Category	Description
Access Control	Evaluates the effectiveness of mechanisms controlling access to ensure only authorized entities can execute specific actions, critical for maintaining system integrity and preventing unauthorized use.
Arithmetic	Focuses on the correct implementation of arithmetic operations to prevent vulnerabilities like overflows and underflows, ensuring that mathematical operations are both logically and semantically accurate.
Complexity	Assesses code organization and function clarity to confirm that functions and modules are organized for ease of understanding and maintenance, thereby reducing unnecessary complexity and enhancing readability.
Data Validation	Assesses the robustness of input validation to prevent common vulnerabilities like overflow, invalid addresses, and other malicious input exploits.
Decentralization	Reviews the implementation of decentralized governance structures to mitigate insider threats and ensure effective risk management during contract upgrades.
Documentation	Reviews the comprehensiveness and clarity of code documentation to ensure that it provides adequate guidance for understanding, maintaining, and securely operating the codebase.
External Dependencies	Evaluates the extent to which the codebase depends on external protocols, oracles, or services. It identifies risks posed by these dependencies, such as compromised data integrity, cascading failures, or reliance on centralized entities. The assessment checks if these external integrations have appropriate fallback mechanisms or redundancy to mitigate risks and protect the protocol's functionality.
Error Handling	Reviews the methods used to handle exceptions and errors, ensuring that failures are managed gracefully and securely.
Logging and Monitoring	Evaluates the use of event auditing and logging to ensure effective tracking of critical system interactions and detect potential anomalies.
Low-Level Calls	Reviews the use of low-level constructs like inline assembly, raw `call` or `delegatecall`, ensuring they are justified, carefully implemented, and do not compromise contract security.

Category	Description
Testing and Verification	Reviews the implementation of unit tests and integration tests to verify that codebase has comprehensive test coverage and reliable mechanisms to catch potential issues.

4.2.1 Rating Criteria

Rating	Description
Excellent	The system is flawless and surpasses standard industry best practices.
Good	Only minor issues were detected; overall, the system adheres to established best practices.
Fair	Issues were identified that could potentially compromise system integrity.
Poor	Numerous issues were identified that compromise system integrity.
Absent	A critical component is absent, severely compromising system safety.
Not Applicable	This category does not apply to the current evaluation.

4.3 FINDINGS CLASSIFICATION REFERENCE

4.3.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

Title	Description
CRITICAL	Issues that pose immediate and significant risks, potentially leading to asset theft, inaccessible funds, unauthorized transactions, or other substantial financial losses. These vulnerabilities represent serious flaws that could be exploited to compromise or control the entire contract. They require immediate attention and remediation to secure the system and prevent further exploitation.
MAJOR	Issues that could cause a significant failure in the contract's functionality, potentially necessitating manual intervention to modify or replace the contract. These vulnerabilities may result in data corruption, malfunctioning logic, or prolonged downtime, requiring substantial operational changes to restore normal performance. While these issues do not immediately lead to financial losses, they compromise the reliability and security of the contract, demanding prioritized attention and remediation.
WARNING	Issues that might disrupt the contract's intended logic, affecting its correct functioning or making it vulnerable to Denial of Service (DDoS) attacks. These problems may result in the unintended triggering of conditions, edge cases, or interactions that could degrade the user experience or impede specific operations. While they do not pose immediate critical risks, they could impact contract reliability and require attention to prevent future vulnerabilities or disruptions.
INFO	Issues that do not impact the security of the project but are reported to the client's team for improvement. They include recommendations related to code quality, gas optimization, and other minor adjustments that could enhance the project's overall performance and maintainability.

4.3.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

Title	Description
NEW	Waiting for the project team's feedback.

Title	Description
FIXED	Recommended fixes have been applied to the project code and the identified issue no longer affects the project's security.
ACKNOWLEDGED	The project team is aware of this finding and acknowledges the associated risks. This finding may affect the overall security of the project; however, based on the risk assessment, the team will decide whether to address it or leave it unchanged.
NO ISSUE	Finding does not affect the overall security of the project and does not violate the logic of its work.

4.4 ABOUT OXORIO

OXORIO is a blockchain security firm that specializes in smart contracts, zk-SNARK solutions, and security consulting. With a decade of blockchain development and five years in smart contract auditing, our expert team delivers premier security services for projects at any stage of maturity and development.

Since 2021, we've conducted key security audits for notable DeFi projects like Lido, 1Inch, Rarible, and deBridge, prioritizing excellence and long-term client relationships. Our co-founders, recognized by the Ethereum and Web3 Foundations, lead our continuous research to address new threats in the blockchain industry. Committed to the industry's trust and advancement, we contribute significantly to security standards and practices through our research and education work.

Our contacts:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

THANK YOU FOR CHOOSING