# FATHOM STABLECOIN

OXORIO

JANUARY 19, 2024

# CONTENTS

# 1 INTRO

# 1.1 DISCLAIMER

The audit makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

# 1.2 ABOUT OXORIO

Oxorio is a young but rapidly growing audit and consulting company in the field of the blockchain industry, providing consulting and security audits for organizations from all over the world. Oxorio has participated in multiple blockchain projects during which smart contract systems were designed and deployed by the company.

Oxorio is the creator, maintainer, and major contributor of several blockchain projects and employs more than 5 blockchain specialists to analyze and develop smart contracts.

Our contacts:

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

# 1.3 SECURITY ASSESSMENT METHODOLOGY

A group of auditors is involved in the work on this audit. Each of them checks the provided source code independently of each other in accordance with the security assessment methodology described below:

**1. Project architecture review**

Study the source code manually to find errors and bugs.

**2. Check the code for known vulnerabilities from the list**

Conduct a verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

**3. Architecture and structure check of the security model**

Study the project documentation and its comparison against the code including the study of the comments and other technical papers.

**4. Result's cross-check by different auditors**

Normally the research of the project is done by more than two auditors. This is followed by a step of mutual cross-check process of the audit results between different task performers.

**5. Report consolidation**

Consolidation of the audited report from multiple auditors.

**6. Reaudit of new editions**

After the provided review and fixes from the client, the found issues are being double-checked. The results are provided in the new version of the audit.

**7. Final audit report publication**

The final audit version is provided to the client and also published on the official website of the company.

# 1.4 FINDINGS CLASSIFICATION

## 1.4.1 Severity Level Reference

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

- **CRITICAL**: A bug leading to assets theft, locked fund access, or any other loss of funds due to transfer to unauthorized parties.
- **MAJOR**: A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- **WARNING**: A bug that can break the intended contract logic or expose it to DDoS attacks.
- **INFO**: Minor issue or recommendation reported to / acknowledged by the client's team.

## 1.4.2 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:

- **NEW**: Waiting for the project team's feedback.
- **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. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.
- **NO ISSUE**: Finding does not affect the overall security of the project and does not violate the logic of its work.
- **DISMISSED**: The issue or recommendation was dismissed by the client.

# 1.5 PROJECT OVERVIEW

Fathom is a decentralized, community governed protocol. Locking FTHM tokens in DAO vault will allow you to put forward proposals and vote on them.
Fathom is a borrow & earn platform where users can stake XDC and tokenized real-world assets (RWA) as collateral to borrow the over-collateralized price stable currency FXD. Fathom is the largest lending/borrowing platform built on XDC Chain. Fathom uses liquid staking and a borrowed interest APR as the foundation to provide users with a variable yet sustainable, high yield for contributing to liquidity pools.

## 1.6 AUDIT SCOPE

The scope of this audit includes smart contracts at the main folder excluding `StableSwapModule` and `StableSwapModuleWrapper`.

The audited commit identifier is `3768c87367d286ae0e82f444b2f9d760417b507e`.

# 2 FINDINGS REPORT

OXORIO

# 2.1 CRITICAL

| | |
|---|---|
| C-01 | Possibility of misconfiguration in `FathomProxyWalletOwner` |
| Severity | CRITICAL |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > `constructor` | 56 |

## Description

In the `constructor` of the contract `FathomProxyWalletOwner`, several `address` type variables, such as `bookKeeper`, `positionManager`, `collateralTokenAdapter`, `stablecoinAdapter`, and so on, are initialized. These variables represent the contracts of the core structure of the protocol. While these contracts are assumed to work as a completely synchronized set, nothing prevents a user from initializing a wallet with the addresses of the contracts that are not synced with each other, like using an address of some obsolete version of one of the contracts.
In such a case, the wallet contract may work incorrectly, possibly resulting in a lock of funds.

## Recommendation

We recommend having a factory or clones contract for the deployment of the `FathomProxyWalletOwner` contract or utilizing an external call to the Configuration contract with all addresses.

## Update

### Client's response

Thanks for the finding. We would like to exclude `FathomProxyWalletOwner` from the audit scope.

| C-02 | Configuration addresses are not updatable in `FathomProxyWalletOwner` |
| --- | --- |
| Severity | CRITICAL |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
| --- | --- | --- |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > `constructor` | 63 |

## Description

In the `constructor` of the contract `FathomProxyWalletOwner`, several `address` type variables, such as `bookKeeper`, `positionManager`, `collateralTokenAdapter`, `stablecoinAdapter`, and so on, are initialized. These variables represent the contracts of the core structure of the protocol. If some of the addresses will get updated within the contracts of the core system, the wallet will not be able to interact with the updated contracts, such as in the case of the call to `setCollateralPoolConfig` in the `BookKeeper` contract. The owner of the wallet won't be able to update already initialized addresses, which may result in a lock of the funds in the protocol.

## Recommendation

We recommend adjusting the architecture for changing main contracts of the protocols by implementing a configuration contract with all recent addresses of the protocol and adding a migration mechanism.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` from the audit scope.

| | |
|---|---|
| C-03 | Missing overflow checks in `PluginPriceOracle` |
| Severity | CRITICAL |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| PluginPriceOracle.sol | contract `PluginPriceOracle` > function `getPrice` | 43 |

## Description

In the function `getPrice` of the contract `PluginPriceOracle`, there is a `latestAnswer` call to the `oracle` contract, which returns the `int256` variable. This `int256` variable is casted to the `uint256` type. Since the received variable from the `oracle` contract is of type `int256`, it can be negative, and by casting a negative variable into the `uint256` type, underflow occurs. As an example: when the oracle is returning `-1`, the result after the `uint256` type casting is `115792089237316195423570985008687907853269984665640564039457584007913129639935`, and after the conversion, the function will revert in the `_toWad` function call because this big underflowed variable with the multiplication to `10**14` will lead to overflow. The overflow with the multiplication will revert because with the Solidity version `0.8.0` and later, there is checked math which protects from having overflow and underflow issues with arithmetic. Despite the fact that revert will occur in the price oracle contract, the overall architecture of the Fathom protocol will handle this situation; however, it's possible that the `latestAnswer` call returns a big negative value, which after the casting and underflow will lead to a smaller `price` that won't revert during the `_toWad` call and later during the calculations. This will lead to the inflated price of the collateral in the system, inability to perform liquidations, and other unexpected behavior.

## Recommendation

We recommend validating the return variable of the `latestAnswer` function call to be a positive value.

## Update

### Client's response

We would like to exclude `PluginPriceOracle` from the audit scope.

## 2.2 MAJOR

| | |
|---|---|
| M-01 | Owner fails to receive native token transfer in `FathomProxyWalletOwner` |
| Severity | MAJOR |
| Status | • ACKNOWLEDGED |

### Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `closePositionFull` | 139 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `withdrawXDC` | 154 |

### Description

In the function `closePositionFull` and the function `withdrawXDC`, the token transfer is performed through the `call` operation to the `msg.sender` cast to `payable` type. The modifier `onlyOwner` ensures that `msg.sender` is the owner of the contract.
The assumption that the owner is always able to receive the transfer may be broken, as in the case when ownership is transferred to the contract without the `receive` method. This will result in the inability to close the position or withdraw native tokens from the protocol.

### Recommendation

We recommend implementing the ERC165 interface to ensure that the owner of the contract is able to receive native tokens.

### Update

#### Client's response

We would like to exclude `PluginPriceOracle` from the audit scope.

| | |
|---|---|
| M-02 | Missing `setOwner` function in `FathomProxyWalletOwner`, `FathomProxyWalletOwnerUpgradeable` |
| Severity | MAJOR |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `buildProxyWallet` | 72 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `buildProxyWallet` | 71 |

## Description

In the function `buildProxyWallet` of the contracts `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable`, in the `build` call to the `ProxyWalletRegistry` contract, the `_owner` address is passed as `address(this)`. This means that the `FathomProxyWalletOwner` is a direct owner of the `proxyWallet` contract in the `ProxyWalletRegistry` contract. However, if the EOA address, which is the owner of the `FathomProxyWalletOwner` contract, decides to change ownership and migrate to another address, it won't be possible. The EOA address is not an owner of the deployed proxy in the `ProxyWalletRegistry` contract, and the call to the `setOwner` function will revert. At the same time, the `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` contracts are missing the `setOwner` function implementation.

## Recommendation

We recommend reviewing the existing logic, adding an implementation for changing the ownership of the proxy in the `FathomProxyWalletOwnerUpgradeable` and `FathomProxyWalletOwner` contracts.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| M-03 | Add migration mechanism in `CollateralPoolConfig` |
| Severity | MAJOR |
| Status | • NO ISSUE |

## Location

| File | Location | Line |
|---|---|---|
| CollateralPoolConfig.sol | contract `CollateralPoolConfig` | 173 |

## Description

The function `setAdapter` in the contract `CollateralPoolConfig` updates the adapter reference in the contract. At the same time, other contracts are not upgraded with this change, which will lead to the usage of different contracts in the protocol and result in unexpected behavior. For example, with the update of the `adapter` in the `CollateralPoolConfig` contract, the `adapter` is not updated in the `FlashMintModule` contract.

## Recommendation

We recommend adding a migration mechanism that accounts for all side effects of updating contracts.

## Update

### Client's response

Fix No Need

Major 3 is about, what happens to `FlashMintModule` if `collateralTokenAdapter` address changes via `CollateralPoolConfig`, But `FlashMintModule` doesn't use `collateralTokenAdapter` but only uses `stablecoinAdapter`.
There are also two more address setter fns in the `CollateralPoolConfig` contract. They are `setPriceFeed` fn and `setStrategy` fn. Contract that uses `priceFeedAddress` of a specific `collateralPool` does not save the `priceFeedAddress` in storage but fetches `priceFeedAddress` from `CollateralPoolConfig` contract. Therefore, `setPriceFeed` in `collateralPoolConfig` does not affect other contracts as was concerned in the issue.
Same for `setStrategy`.

# 2.3 WARNING

| | |
|---|---|
| W-01 | Addresses are not validated in `FathomProxyWalletOwner` |
| Severity | WARNING |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > `constructor` | 47 |

## Description

In the `constructor` of the contract `FathomProxyWalletOwner`, the addresses of the protocol contracts are supplied by the user. While the addresses get validated for being empty, no validation is performed that those addresses represent correct entities of the protocol or represent contracts at all. The interfaces are not ensured for supplied addresses.

## Recommendation

We recommend deploying `FathomProxyWalletOwner` as a clone or using a factory contract, or validating addresses to represent legitimate instances of the protocol. The protocol can benefit from the architecture with a centralized configuration contract. Using ERC165 to validate interface implementation is advised.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` from the audit scope.

| | |
|---|---|
| W-02 | DDOS attack in `FathomProxyWalletOwnerUpgradeable` |
| Severity | WARNING |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `receive` | 37 |

## Description

In the function `receive` of the contract `FathomProxyWalletOwnerUpgradeable`, the event is emitted notifying about received funds. The entities subscribed to this event may experience denial of service in case a malicious actor would spam the contract with transfers of negligible amounts (1 wei) of native tokens. This can cause unexpected behavior for integrators using the `FathomProxyWalletOwnerUpgradeable` contract.

## Recommendation

We recommend removing the `emit` or notifying all external integrators not to use this event for production monitoring; this `emit` should be used only for checking historical values.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| W-03 | Disable initializers in upgradable contracts |
| Severity | WARNING |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` | 17 |
| ProxyWalletRegistry.sol | contract `ProxyWalletRegistry` | 39 |

## Description

The contract `FathomProxyWalletOwnerUpgradeable` and the contract `ProxyWalletRegistry` are upgradable, inheriting from the `Initializable` contract. However, the current implementation is missing the `_disableInitializers` function call in the constructor. Thus, an attacker can initialize the implementation. Usually, the initialized implementation has no direct impact on the proxy itself; however, it can be exploited in a phishing attack. In rare cases, the implementation might be mutable and may have an impact on the proxy.
Same applies to other upgradable contracts in the protocol.

## Recommendation

We recommend calling `_disableInitializers` within the contract's constructor to prevent the implementation from being initialized.

## Update

### Client's response

Fixed in commit `657749cc9c6669c0c388588564e4dea946feed35`.
Fixed as suggested. added

```
constructor() {
    _disableInitializers();
}
```

To upgradeable contracts so that `_disableInitializers` will be called at the moment of implementation deployment. Thanks for the advice.

| | |
|---|---|
| W-04 | Received amount of stablecoin is not validated in `FathomProxyWalletOwnerUpgradeable` |
| Severity | WARNING |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `openPosition` | 91 |

## Description

In the function `openPosition` of the contract `FathomProxyWalletOwnerUpgradeable`, the transferred stablecoin amount equals the balance of the contract. This amount is not verified to match the parameter `_stablecoinAmount`. The function will not fail even if the factually transferred amount will be less than the requested `_stablecoinAmount` or even if no tokens will be transferred at all.

## Recommendation

We recommend verifying that the transferred amount matches the requested amount.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| W-05 | Casting to types in unsafe way |
| Severity | WARNING |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| CollateralTokenAdapter.sol | contract `CollateralTokenAdapter` > function `_deposit` | 189 |

## Description

In these locations, there are casts to the `int256` and `uint256` types. In some cases, there are validations to prevent overflow, while in other locations, any checks are missing.

## Recommendation

We recommend unifying the handling of casting to `int256` by utilizing `safeToInt256` and `safeToUint256` functions in all places, validating scenarios when the variable can overflow/underflow.

## Update

### Client's response

Fixed in commit `e368d26bf75e871e983bbd7bd60f446e92ce2396`.
Fixed as suggested.

Added `safeToInt256` and `safeToUint256` to the type casting in contracts. Left `CollateralTokenAdapter`'s casting with overflow/underflow check as is.

| | |
|---|---|
| W-06 | `feeRate` is not limited in `FlashMintModule` |
| Severity | WARNING |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| FlashMintModule.sol | contract `FlashMintModule` > function `setFeeRate` | 122 |

## Description

In the function `setFeeRate` of the contract `FlashMintModule`, the `feeRate` is unbounded when it's getting set. The `feeRate` variable can be set greater than `WAD`, which will lead to the accumulation of fees greater than the actual flash loan amount.

## Recommendation

We recommend validating the `feeRate` value.

## Update

### Client's response

Fixed in commit `a23bebf1b097aec1754ed692f87b8d31e239ec2f`.
Fixed as suggested.

set limitation to `feeRate`. The `feeRate` cannot be higher than `WAD` since `WAD` is 100%.

| | |
|---|---|
| W-07 | Redundant check for `totalStablecoinIssued` in multiple contracts |
| Severity | WARNING |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| PositionManager.sol | contract `PositionManager` > function `setBookKeeper` | 333 |
| PositionManager.sol | contract `PositionManager` > function `initialize` | 85 |
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` > function `initialize` | 110 |
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` > function `setBookKeeper` | 258 |
| LiquidationEngine.sol | contract `LiquidationEngine` > function `initialize` | 95 |
| LiquidationEngine.sol | contract `LiquidationEngine` > function `setBookKeeper` | 212 |
| PriceOracle.sol | contract `PriceOracle` > function `setBookKeeper` | 84 |
| PriceOracle.sol | contract `PriceOracle` > function `initialize` | 77 |
| ShowStopper.sol | contract `ShowStopper` > function `initialize` | 65 |
| ShowStopper.sol | contract `ShowStopper` > function `setBookKeeper` | 72 |

## Description

In the mentioned locations, the `require` statement that checks `totalStablecoinIssued >= 0` is redundant since `totalStablecoinIssued` is of type `uint256`.

## Recommendation

We recommend removing the redundant check.

## Update

### Client's response

Fixed in commit `f024a0f23a8ad9975a1651d6a5beb7dffef74629`.
Fixed as suggested.
Redundant checks for `totalStablecoinIssued` in multiple contracts removed.

| | |
|---|---|
| W-08 | Validation of `_totalDebtCeiling` in `BookKeeper` |
| Severity | WARNING |
| Status | • NO ISSUE |

## Location

| File | Location | Line |
|---|---|---|
| BookKeeper.sol | contract `BookKeeper` > function `setTotalDebtCeiling` | 150 |

## Description

In the function `setTotalDebtCeiling` of the contract `BookKeeper`, the `_totalDebtCeiling` parameter must be passed in rad, which is not enforced. Incorrect variable passed will lead to the failed calls of the `adjustPosition` transactions in the `BookKeeper`.

## Recommendation

We recommend validating the parameter to be passed in rad.

## Update

### Client's response

Fix no need.
The `debtCeiling` can be theoretically 0.5 FXD, which is below 1 `RAD` of debt ceiling.

| | |
|---|---|
| W-09 | Validation of `_debtFloor` in `CollateralPoolConfig` |
| Severity | WARNING |
| Status | • NO ISSUE |

## Location

| File | Location | Line |
|---|---|---|
| CollateralPoolConfig.sol | contract `CollateralPoolConfig` > function `setDebtFloor` | 118 |

## Description

In the function `setDebtFloor` of the contract `CollateralPoolConfig`, the parameter `_debtFloor` must be passed in rad, which is not enforced.

## Recommendation

We recommend validating the parameter to be passed in rad.

## Update

### Client's response

Fix no need.
Position debt floor can theoretically be 0.5 FXD or even 0, when debt floor is not forced.

| | |
|---|---|
| W-10 | Lack of sanity check in `CollateralPoolConfig` |
| Severity | WARNING |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| CollateralPoolConfig.sol | contract `CollateralPoolConfig` > function `setStrategy` | 212 |

## Description

In the `setStrategy` function of the `CollateralPoolConfig` contract, there is no sanity check call to validate if the provided `strategy` has the correct interface. An incorrect `strategy` address will lead to failed liquidation transactions.

## Recommendation

We recommend verifying the `strategy` address by incorporating a sanity check call to the `flashLendingEnabled` variable.

## Update

### Client's response

The described concern is understandable. However, there is not much utility, at the moment, in making the change since checking whether `flashLendingEnabled` return bool or not doesn't necessarily check if the new `fixedSpreadLiquidationStrategy` is really valid or not.

| | |
|---|---|
| W-11 | Outdated typing in `FixedSpreadLiquidationStrategy` |
| Severity | WARNING |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` | 56 |
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` > function `setFlashLendingEnabled` | 143 |

## Description

In the `FixedSpreadLiquidationStrategy` contract, the `flashLendingEnabled` variable is incorrectly defined as `uint256` instead of the appropriate `bool` type.

The same issue exists in the `setFlashLendingEnabled` function, where the parameter `_flashLendingEnabled` is erroneously defined as `uint256`.

## Recommendation

We recommend changing the variable type to `bool`.

## Update

### Client's response

Fixed in commit `11066b49d5b632a66b495d437031c6969b65558f`.
Fixed as suggested

`flashLendingEnabled` from `uint` to `bool`.
event, function, execute fn's subroutine all changed accordingly.

## 2.4 INFO

| | |
|---|---|
| I-01 | Use of outdated libraries |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

### Location

| File | Location | Line |
|---|---|---|
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` > function `execute` | 202 |

### Description

The protocol currently employs the following versions of the OpenZeppelin libraries:

```
"@openzeppelin/contracts": "4.4.1",
"@openzeppelin/contracts-upgradeable": "4.4.1",
```

These versions are considered outdated, with the current version of the OpenZeppelin library being `v5.0.1`. The use of outdated libraries increases the risk of vulnerabilities in the code. According to SNYK, the current version of the library (v5.0.1) addresses multiple High and Medium issues. For instance, in the `execute` function of the `FixedSpreadLiquidationStrategy` contract, there is a `supportsInterface` call that can consume excessive resources when processing a large amount of data via an EIP-165 and revert, potentially leading to a Denial of Service (DoS) attack. While this issue doesn't directly impact the `FixedSpreadLiquidationStrategy` itself, the `_collateralRecipient` could be affected. Consequently, it is strongly recommended to update the OpenZeppelin package.

### Recommendation

We recommend updating the OpenZeppelin package to the latest version (v5.0.1).

## Update

### Client's response

The concern raised is acknowledged and understood. While upgrading to the recommended version 5.0.1 of OpenZeppelin (OZ) would be ideal, such an upgrade from our current version 4.4.1 to 5.0.1 necessitates substantial changes to the codebase. Therefore, at this stage, we have opted to update the OZ version from 4.4.1 to 4.9.2. This interim upgrade has been implemented in commit `fa337f9778480a55d4c24274b9c315a55952e308`.

| | |
|---|---|
| I-02 | Gas consumption limitations for integrators in `FathomStablecoinProxyActions` |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomStablecoinProxyActions.sol | contract `FathomStablecoinProxyActions` > function `wipeAndUnlockXDC` | 143 |

## Description

In the function `wipeAndUnlockXDC` of the `FathomStablecoinProxyActions` contract, there is a gas consumption limitation in the `safeTransferETH` call. This limitation can result in a failed transaction if the integrator is using the `receive` function with custom and gas-heavy logic.

## Recommendation

We recommend a thorough review of the existing logic.

## Update

### Client's response

Described concern is understandable. The intention of limiting the gas amount to 21,000 was to ensure that the gas is only sufficient for the `ETH` transfer and nothing more.

| | |
|---|---|
| I-03 | Unsafe usage of `abi.encodeWithSelector` in `SafeToken` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| SafeToken.sol | contract `SafeToken` > function `safeTransfer` | 27 |

## Description

In the function `safeTransfer` of the `SafeToken` contract, `abi.encodeWithSelector` is used instead of `abi.encodeCall`. Since version `0.8.11`, `abi.encodeCall` provides a type-safe encoding utility compared to `abi.encodeWithSelector`. While `abi.encodeWithSelector` can be used with `interface.<function>.selector` to prevent typographical errors, it lacks type checking during compile time, a feature offered by `abi.encodeCall`.

## Recommendation

We recommend using `abi.encodeCall` instead of `abi.encodeWithSelector` to adhere to best practices in the web3 sphere.

## Update

### Client's response

Fixed in commit `327df2174f5c9514b3dc2c692adde3e5a293432e`.
Fixed as suggested.

| | |
|---|---|
| I-04 | Impossible to withdraw partial amount |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `withdrawStablecoin` | 144 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `withdrawXDC` | 151 |

## Description

The functions `withdrawStablecoin` and `withdrawXDC` in the `FathomProxyWalletOwner` contract withdraw funds using the `balanceOf(address(this))` statement, allowing withdrawal only of the full balance of the contract.
Same applies to the upgradeable version of the contract.

## Recommendation

We recommend allowing partial withdrawal by providing the `amount` parameter to the `withdrawStablecoin` and `withdrawXDC` functions.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| I-05 | Transfer amount validation in `FathomProxyWalletOwnerUpgradeable` |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `openPosition` | 90 |

## Description

In the function `openPosition`, the `msg.value` amount in the `FathomProxyWalletOwnerUpgradeable` contract is not validated to be non-zero.
Same applies to the non-upgradeable version of the contract.

## Recommendation

We recommend validating that `msg.value` is positive before performing any logic related to opening a position.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| I-06 | Check is not performed prior to sending funds in `FathomProxyWalletOwner` |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `closePositionPartial` | 114 |

## Description

In the function `closePositionPartial` of the contract `FathomProxyWalletOwner`, the `wipeAndUnlockXDC` call sends native tokens to the `FathomProxyWalletOwner` only when `_collateralAmount > 0`. However, the following call:

```
(bool sent, ) = payable(msg.sender).call{ value: address(this).balance }("");
```

which sends the whole balance of the contract, is always executed.
Same applies to the upgradeable version of the contract.

## Recommendation

We recommend executing the transfer of the balance only if funds were received after the `wipeAndUnlockXDC` function call.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| I-07 | Validation is redundant in `FathomProxyWalletOwner`, `FathomProxyWalletOwnerUpgradeable` |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `closePositionFull` | 122 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `ownerFirstPositionId` | 161 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `ownerLastPositionId` | 167 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `ownerPositionCount` | 173 |
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` > function `list` | 180 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `closePositionFull` | 122 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `ownerFirstPositionId` | 161 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `ownerLastPositionId` | 167 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `ownerPositionCount` | 173 |
| FathomProxyWalletOwnerUpgradeabl... | contract `FathomProxyWalletOwnerUpgradeable` > function `list` | 180 |

## Description

In the mentioned locations of the contract `FathomProxyWalletOwner` and the contract `FathomProxyWalletOwnerUpgradeable` the validation of variables for zero-value is redundant as the prior call `_validateAddress(proxyWallet)` passes only if `proxyWallet` was initialized, which is possible only after all the other contracts were initialized in the constructor or initializer.

## Recommendation

We recommend removing the redundant validation to keep the codebase clean.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| | |
|---|---|
| I-08 | Typo in `PositionManager` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| PositionManager.sol | contract `PositionManager` | 255 |

## Description

In the contract `PositionManager`, there is a typo in the comment in the word `addresss`.

## Recommendation

We recommend fixing the typo to `address`.

## Update

### Client's response

Fixed in commit `96a66f914b433d1535eee546ff3f1da80055e6d2`.
Fixed as suggested.

| | |
|---|---|
| I-09 | Redundant Imports |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| PositionManager.sol | contract `PositionManager` | 9 |
| CentralizedOraclePriceFeed.sol | contract `CentralizedOraclePriceFeed` | 6 |
| SlidingWindowDexOracle.sol | contract `SlidingWindowDexOracle` | 8 |
| FathomStablecoinProxyActions.sol | contract `FathomStablecoinProxyActions` | 11 |
| FathomStablecoinProxyActions.sol | contract `FathomStablecoinProxyActions` | 12 |
| CollateralTokenAdapter.sol | contract `CollateralTokenAdapter` | 10 |
| AdminControls.sol | contract `AdminControls` | 7 |
| ShowStopper.sol | contract `ShowStopper` | 12 |
| SystemDebtEngine.sol | contract `SystemDebtEngine` | 9 |

## Description

There are redundant imports in the mentioned locations.

## Recommendation

We recommend removing redundant imports to keep the codebase clean.

## Update

### Client's response

Fixed in commit `d4f064853cd9c1085d0fd74ebedfdeb5df3a2903`.
Fixed as suggested.

| | |
|---|---|
| I-10 | Obsolete comments in `PositionManager` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| PositionManager.sol | contract `PositionManager` > function `allowManagePosition` | 98 |

## Description

In the contract `PositionManager`, the comment for the function `allowManagePosition` suggests obsolete typing for the `_ok` parameter, which is of `bool` type.

## Recommendation

We recommend fixing the comment to match the variable type.

## Update

### Client's response

Fixed in commit `5fe984f2ed654ae9c4621ca4dddbbbe8c04ca936`.
Fixed as suggested.

| | |
|---|---|
| I-11 | `Address` imported instead of `AddressUpgradeable` in `BookKeeper` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| BookKeeper.sol | None | 6 |

## Description

In the contract `BookKeeper`, the `Address` contract is imported, while other OpenZeppelin imports use the upgradeable version of the contracts.

## Recommendation

We recommend importing the `AddressUpgradeable` contract to unify imports.

## Update

### Client's response

Fixed in commit `08eeb18486d2fe95ea5e06d4383e5f817b0fba45`.
Fixed as suggested.

| | |
|---|---|
| I-12 | Typo in error message in `FixedSpreadLiquidationStrategy` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| FixedSpreadLiquidationStrategy.sol | contract `FixedSpreadLiquidationStrategy` > function `execute` | 165 |

## Description

In the function `execute` of the contract `FixedSpreadLiquidationStrategy`, the error message contains a typo - `liquidationEngingRole`.

## Recommendation

We recommend replacing the error message with `LIQUIDATION_ENGINE_ROLE`.

## Update

### Client's response

Fixed in commit `55e02025dcfc9a0e90deb5b1d6ba644e07452a1a`.
Fixed as suggested.

| | |
|---|---|
| I-13 | Missing `require` check in `FlashMintModule` |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
|---|---|---|
| FlashMintModule.sol | contract `FlashMintModule` > function `flashLoan` | 162 |

## Description

In the function `flashLoan` of the contract `FlashMintModule`, there is no validation that after the `settleSystemBadDebt` call, the current amount of the `stablecoin` in `BookKeeper` equals or is greater than the previous amount plus fees, while this check is present in the `bookKeeperFlashLoan` function.

## Recommendation

We recommend adding the same `require` to the `flashLoan` function to ensure the same level of security in both functions.

## Update

### Client's response

Fixed in commit `754fd35abc2c6ea77e5e4e375fb587b8fcf114a6`.
Fixed as suggested.

| | |
|---|---|
| I-14 | Typo in `FathomProxyWalletOwner` |
| Severity | INFO |
| Status | • ACKNOWLEDGED |

## Location

| File | Location | Line |
|---|---|---|
| FathomProxyWalletOwner.sol | contract `FathomProxyWalletOwner` | 207 |

## Description

In the contract `FathomProxyWalletOwner`, there is a typo in the function name `_successfullXDCTransfer` instead of `_successfulXDCTransfer`.

## Recommendation

We recommend correcting the typo to `_successfulXDCTransfer` for consistency and clarity.

## Update

### Client's response

We would like to exclude `FathomProxyWalletOwner` and `FathomProxyWalletOwnerUpgradeable` from the audit scope.

| I-15 | Typo in several contracts |
| --- | --- |
| Severity | INFO |
| Status | • FIXED |

## Location

| File | Location | Line |
| --- | --- | --- |
| IDelayPriceFeed.sol | interface `IDelayPriceFeed` | 23 |
| CentralizedOraclePriceFeed.sol | contract `CentralizedOraclePriceFeed` | 33 |
| DelayFathomOraclePriceFeed.sol | contract `DelayFathomOraclePriceFeed` | 54 |
| DelayPriceFeedBase.sol | contract `DelayPriceFeedBase` > function `peekPrice` | 73 |
| DelayPriceFeedBase.sol | contract `DelayPriceFeedBase` | 103 |

## Description

In mentioned locations “retrive” in the names of functions should be replaced with “retrieve”.

## Recommendation

We recommend fixing the typo.

## Update

### Client's response

Fixed in commit `aab9f1a307e3d8e238f54913a350ca74bcba446c`.
Fixed as suggested.

# 3 CONCLUSION

The following table contains the total number of issues that were found during audit:

| Severity | FIXED | ACKNOWLEDGED | NO ISSUE | Total |
|---|---|---|---|---|
| CRITICAL | 0 | 3 | 0 | 3 |
| MAJOR | 0 | 2 | 1 | 3 |
| WARNING | 5 | 4 | 2 | 11 |
| INFO | 8 | 7 | 0 | 15 |
| Total | 13 | 16 | 3 | 32 |

THANK YOU FOR CHOOSING