Quantstamp Security Assessment Certificate

December 23rd 2022 — Quantstamp Verified

Hifi Finance

This audit report was prepared by Quantstamp, the leader in blockchain security.

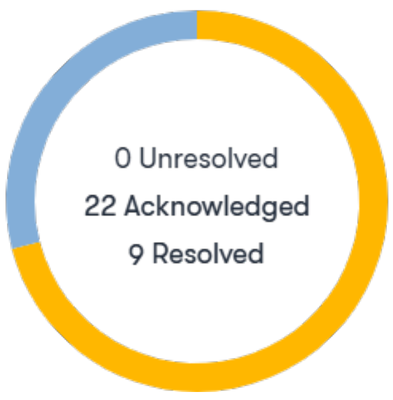
Executive Summary

Type	DeFi protocol
Auditors	Souhail Mssassi, Research Engineer Zeeshan Meghji, Auditing Engineer Roman Rohleder, Research Engineer
Timeline	2022-10-05 through 2022-10-21
EVM Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Hifi Docs
Documentation Quality	Medium
Test Quality	Medium

Source Code

Repository	Commit
hifi-finance/hifi	a48417d initial audit
hifi-finance/hifi	b9282b6 fixes

Total Issues	**31** (9 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**15** (6 Resolved)
Low Risk Issues	**6** (2 Resolved)
Informational Risk Issues	**6** (1 Resolved)
Undetermined Risk Issues	**4** (0 Resolved)



High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Initial audit:
Through reviewing the code, we found **31 potential issues** of various levels of severity: 15 medium-severity, 6 low-severity, 6 informational-severity, 4 undetermined-severity. We recommend addressing all the issues before deploying the code.

ID	Description	Severity	Status
QSP-1	Denial of Service on Repayments and Liquidations	^ Medium	Acknowledged
QSP-2	Smaller Liquidity Providers Can Be Priced Out	^ Medium	Acknowledged
QSP-3	Debt Celing Can Be Exceeded	^ Medium	Acknowledged
QSP-4	Can Redeem Even when `Fintroller` Disallows Redemption	^ Medium	Fixed
QSP-5	Insufficient Slippage Protection for Liquidity Operations	^ Medium	Acknowledged
QSP-6	`depositUnderlying()` Can Result in Temporarily Stuck Funds	^ Medium	Fixed
QSP-7	Insufficient Price Feed Validation	^ Medium	Fixed
QSP-8	Inconsistent Configuration on `Fintroller`	^ Medium	Fixed
QSP-9	Unbounded Iteration on Collateral Assets	^ Medium	Mitigated
QSP-10	Potential Re-Entrancy/Checks-Effects-Interactions Pattern Violations	^ Medium	Acknowledged
QSP-11	Unlimited Approval in `HifiProxyTarget.approveSpender()`	^ Medium	Acknowledged
QSP-12	Front Run Can Lead to Loss in Incentive	^ Medium	Acknowledged
QSP-13	Owner Can Override a Bond	^ Medium	Acknowledged
QSP-14	Usage of `transfer()` Instead of `safeTransfer()`	^ Medium	Fixed
QSP-15	Loss of Precision	^ Medium	Acknowledged
QSP-16	No Storage Gap Declared on `OwnableUpgradeable`	v Low	Fixed
QSP-17	User Can Receive a Discount Due to Loss of Precision	v Low	Acknowledged
QSP-18	Missing Input Validation	v Low	Acknowledged
QSP-19	Privileged Roles and Ownership	v Low	Acknowledged
QSP-20	Unsafe Casts	v Low	Fixed
QSP-21	Return Values Not Verified	v Low	Acknowledged
QSP-22	Events Emitted After External Calls	o Informational	Acknowledged
QSP-23	Unlocked Pragma	o Informational	Mitigated
QSP-24	Ownership Can Be Renounced	o Informational	Acknowledged
QSP-25	Clone-and-Own	o Informational	Acknowledged
QSP-26	Test Code Mixed with Production Code	o Informational	Acknowledged
QSP-27	Transaction Ordering Dependence for `BalanceSheetV2.initialize()`	o Informational	Acknowledged
QSP-28	Minimal/incomplete Pause Mechanism	? Undetermined	Acknowledged
QSP-29	Reliance on Unaudited Contracts and Libraries	? Undetermined	Acknowledged
QSP-30	`ChainlinkOperator` Does Not Allow for Multiple Assets with the Same Symbol	? Undetermined	Acknowledged
QSP-31	Protocol only Works with Tokens with up to 18 Decimals	? Undetermined	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

DISCLAIMER:
Only the following sub-folders were in scope for this audit: `./packages/amm/contracts/*`, `./packages/protocol/contracts/*` and `./packages/proxy-target/contracts/*`.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Denial of Service on Repayments and Liquidations

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`

Description: The `BalanceSheetV2` includes functions for repaying debt (`repayBorrow()`, `repayBorrowBehalf()`) as well as liquidating debt (`liquidateBorrow()`). All these functions will revert if the user attempts to repay more than the vault's current debt. The code below causes a revert when a user attempts to repay more than the vault's debt. Similar logic applies to liquidations.

```
uint256 debtAmount = vaults[borrower].debtAmounts[bond];
if (debtAmount < repayAmount) {
    revert BalanceSheet__RepayBorrowInsufficientDebt(bond, repayAmount, debtAmount);
}
```

The logic above can lead to a denial of service for liquidators or borrowers trying to repay their loans. If a user tries to repay the total amount of their debt, they may be front-run by a malicious liquidator, which calls `repayBorrowBehalf()` with a minimal repayment. The transaction for the normal user will revert because the amount they are attempting to repay exceeds the vault's

debt by a small amount. A similar scenario can occur when a malicious user prevents liquidation by front-running a valid call to `liquidateBorrow()` with either a small repayment or a small liquidation.

Exploit Scenario:

1. A borrower attempts to pay off their entire debt `d` by calling `repayBorrow(bond, d)`
2. A malicious user who wants to liquidate the borrower in a subsequent block front-runs the borrower's call with a call to `repayBorrowBehalf(bond, 1)`. The debt of the borrower's vault is now `d-1`.
3. The borrower's transaction fails because `d > d-1`.
4. The malicious user repeats step 2 until the borrower's position becomes unhealthy. They then liquidate the borrower's vault.

Recommendation: `liquidateBorrow()`, `repayBorrow()`, and `repayBorrowBehalf()` should not revert if the amount of debt offered exceeds the vault's debt. Instead, the function should return any excess payment to the sender. Furthermore, consider either removing the `repayBorrowBehalf()` function or modifying it such that only addresses approved by the borrower can repay their debt. This would ensure that no user who is not explicitly approved by the borrower can interfere with the borrower's repayment.

Update: While some forms of this attack have now been blocked due to the addition of `onlyOwner` to the `repayBorrowBehalf()` function, the attack is still possible through the `liquidateBorrow()`, which may be used to front-run legitimate full liquidations with a trivially small liquidation. The small liquidation would slightly decrease the debt and thus, the full liquidation would fail as it is attempting to repay more than the existing amount of debt. From the team:

> Added “onlyOwner” modifier to “repayBorrowBehalf” to eliminate the risk of borrow repayment being front run. The issue with liquidations being front ran via “liquidateBorrow” still remains, but that is a risk we are willing to accept.

QSP-2 Smaller Liquidity Providers Can Be Priced Out

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`

Description: An early liquidity provider can manipulate a single share's price to raise it high enough such that only the largest providers can afford to add liquidity. The share price can be manipulated by first only adding a tiny amount of liquidity and then directly transferring a large amount of the underlying token to the pool.

Exploit Scenario:

1. The first liquidity provider calls `HifiPool.mint(1)` with `1` Wei of underlying. The liquidity provider receives `1` liquidity token, which is currently the entire supply.
2. The liquidity provider transfers `10**21` of the underlying token directly to the `HifiPool`. The liquidity token supply is still `1`, which means that `1` share is now worth `10**21 + 1` underlying tokens.
3. If any liquidity provider wants to add liquidity, they will need to provide a minimum of `10**21` underlying tokens to obtain just `1` share.

Recommendation: Uniswap V2 solves a similar problem by defining a minimum amount of liquidity tokens (`10**3`) which are permanently locked. This raises the cost of the above attack by a factor of `10**3`.

Update: The Hifi team chose not to resolve the issue as ill-initialized pools can be recreated. From the Hifi team :

> It makes sense to have this fix for a Uniswap V2 pool where the pool contract of tokens A and B could only be created once since it is created via CREATE2, but that is not the case for our protocol. In our protocol, you could always create a new Hifi pool to replace an ill-initialized Hifi pool.

QSP-3 Debt Celing Can Be Exceeded

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`

Description: The `Fintroller` contract defines a debt `ceiling` for every `Bond` added. The following code within the `BalanceSheetV2.borrow()` implies that the total debt is the supply of the hToken corresponding to the bond:

```
uint256 newTotalSupply = bond.totalSupply() + borrowAmount;
uint256 debtCeiling = fintroller.getDebtCeiling(bond);
if (newTotalSupply > debtCeiling) {
    revert BalanceSheet__DebtCeilingOverflow(newTotalSupply, debtCeiling);
}
```

However, it is possible to increase the supply of the hToken past the debt ceiling by calling the `HToken.depositUnderlying()` function. This function mints new hTokens so long as the equivalent amount of underlying is supplied. There are no checks to ensure that hTokens will not be minted past the debt ceiling.

Recommendation: Add validation to the `HToken.depositUnderlying()` function to ensure that the total supply of the hToken remains below the debt ceiling.

Update: The Hifi team indicated that the purpose of the debt ceiling is to minimize risk. They have indicated that risk is not increased by minting HTokens in this way, since they are one-to-one with the underlying. From the team:

> The purpose of the debt ceiling is to mitigate risk, depositing underlying to mint new hTokens does not increase risk because they are minted 1:1.

QSP-4 Can Redeem Even when `Fintroller` Disallows Redemption

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`

Description: The `HToken.redeem()` function allows a user to redeem hTokens in exchange for underlying tokens after maturity. The `Fintroller` contract defines a configuration `isRedeemHTokenAllowed` for every `Bond` added which indicates whether the hToken can be redeemed. However, the `HToken.redeem()` currently allows a user to redeem hTokens even if `isRedeemHTokenAllowed` is false as defined by the `Fintroller`.

Recommendation: Add validation to the `HToken.redeem()` function so that it reverts if `isRedeemHTokenAllowed` is false as defined by the `Fintroller`.

Update: Fixed in commit

3126a78 by acknowledging `isRedeemHTokenAllowed` to be a left over of a previous refactoring and removing it.

QSP-5 Insufficient Slippage Protection for Liquidity Operations

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: The `HifiProxyTarget` features many functions which either add or remove liquidity from the `HifiPool` contract. All of these functions are missing sufficient protection against slippage. Liquidity providers are subject to impermanent loss when the interest rate for the `HifiPool` changes. The interest rate depends on the size of the hToken and underlying reserves. Liquidity providers must provide the underlying tokens and hTokens in their desired expected ratio. They should also always receive the amounts of tokens they expect when removing liquidity if they do not wish to realize an impermanent loss.
There are some functions within the `HifiProxyTarget` which provide some slippage protection such as `addLiquidity()` which requires a `maxHTokenRequired` parameter. This protects the price of the hToken going up. However, there is no protection provided against the price of the underlying token (relative to the hToken) going up. To sufficiently protect against slippage, a `minHTokenRequired` parameter should be used as well. Similarly, functions that remove liquidity, such as `removeLiquidity()`, lack parameters to indicate the minimum amount of hTokens and underlying tokens received. We have listed all functions on the `HifiProxyTarget` which lack sufficient slippage protection:

- `addLiquidity()`
- `addLiquidityWithSignature()`
- `borrowHTokenAndAddLiquidity()`
- `borrowHTokenAndAddLiquidityWithSignature()`
- `buyHTokenAndAddLiquidity()`
- `buyHTokenAndAddLiquidityWithSignature()`
- `buyUnderlyingAndAddLiquidity()`
- `buyUnderlyingAndAddLiquidityWithSignature()`
- `depositCollateralAndBorrowHTokenAndAddLiquidity()`
- `depositCollateralAndBorrowHTokenAndAddLiquidityWithSignature()`
- `depositUnderlyingAndMintHTokenAndAddLiquidity()`
- `depositUnderlyingAndMintHTokenAndAddLiquidityWithSignature()`
- `removeLiquidity()`
- `removeLiquidityAndRedeem()`
- `removeLiquidityAndRedeemWithSignature()`
- `removeLiquidityAndSellHToken()`
- `removeLiquidityAndSellHTokenWithSignature()`
- `removeLiquidityAndWithdrawUnderlying()`
- `removeLiquidityAndWithdrawUnderlyingWithSignature()`
- `removeLiquidityWithSignature()`

Recommendation: All functions within the `HifiProxyTarget` contract which add liquidity must have parameters to indicate the maximum and minimum reserve ratio between the underlying token and the hToken. Functions that remove liquidity must have parameters that indicate the minimum amount of hTokens and underlying tokens to receive from `HifiPool` when burning the liquidity tokens. An example of sufficient slippage protection can be seen in Uniswap V2's functions for adding liquidity and removing liquidity.

Update: The Hifi team has attempted to mitigate the issue by using slippage checks on the front-end. However, front-ends cannot sufficiently protect against slippage due to front-running of transactions.

QSP-6 `depositUnderlying()` Can Result in Temporarily Stuck Funds

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: The `HifiProxyTarget.depositUnderlying()` function is a simple wrapper function around the function `HToken.depositUnderlying()` which receives underlying tokens and sends back hTokens. The `HToken` contract will also internally increment the caller's balance. In this case, the caller will be the DSProxy contract. The `HToken.withdrawUnderlying()` function allows a user to withdraw the underlying tokens they deposited using the `HifiProxyTarget.depositUnderlying()` function. However, the `HifiProxyTarget` contract does not contain an equivalent for the `HifiProxyTarget.withdrawUnderlying()` function. The closes match in the `HifiProxy` contract would be the `HifiProxyTarget.removeLiquidityAndWithdrawUnderlying()` function, which would require a user to additionally add sufficient liquidity to the `HifiPool` contract. The user may not have the funds to add that liquidity, and as a result, they will not be able to withdraw their underlying tokens until maturity. At maturity, the user could call `HifiProxyTarget.redeem()`.

Recommendation: Add a `withdrawUnderlying()` function to the `HifiProxyTarget` contract which simply calls `HToken.WithdrawUnderlying` and relays the underlying tokens back to the user. Alternatively, remove the `HifiProxyTarget.depositUnderlying()` function

Update: Fixed in commit 4c4ae15 by removing function `HifiProxyTarget.depositUnderlying()`, as suggested.

QSP-7 Insufficient Price Feed Validation

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/protocol/contracts/oracles/ChainlinkOperator.sol`

Description: The Hifi protocol is highly dependent on Chainlink price feeds, which are read from the `ChainlinkOperator` contract. While the `getPrice()` function validates that the retrieved price exceeds zero, it does not validate the staleness of the price. Since an attacker could use the predictability of old prices to manipulate the protocol, it is critical that the protocol validates that all prices are fresh.

Recommendation: Validate that the `updatedAt` value returned by `IAggregatorV3.latestRoundData()` indicates a timestamp that is not too old.

Update: Fixed in commit 4ddb307 by a new state variable `priceStalenessThreshold` and checking the returned price freshness to be within its value.

QSP-8 Inconsistent Configuration on `Fintroller`

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`

Description: It is possible to call the configuration setter functions on Fintroller such that the overall configuration for a particular `Bond` becomes inconsistent or invalid. We have listed some of these possibilities below:

1. If borrowing is allowed for a `Bond`, then liquidation should also be allowed for a `Bond`. If liquidation is not allowed, it may allow the bad debt of the protocol to grow unreasonably. It is currently possible to set `isBorrowAllowed` for a Bond to `true` through the `setBorrowAllowed()` function, even when `isLiquidateBorrowAllowed` for the `Bond` is set to `false`.
2. The collateral ratio for a `Collateral` must always be greater than the liquidation incentive. For example, if the liquidation incentive is `10%`, then the minimum collateral ratio should be at least `110%`. However, no such check is made in `setCollateralRatio()`, leaving it possible to set a collateral ratio lower than a collateral's liquidation incentive. The same is true for `setLiquidationIncentive()`, which does not contain the necessary validation checks.
3. If liquidation is allowed for a `Bond`, then repayment should already be allowed. It would be unfair to allow a user's debt to be liquidated without giving them a chance to repay it. However, it is possible to call `setLiquidateBorrowAllowed()` such that liquidation is enabled while repayment is not enabled for a particular `Bond`. Furthermore, it is possible to disable liquidation for a `Bond` even when borrowing is allowed. As mentioned in point `1`, this should not be possible.
4. It should only be possible to disable repayment for a `Bond` if liquidation is also disabled. However, it is possible to disable repayment through the `setRepayBorrowAllowed()` function even when liquidation is still allowed for that `Bond.`

Recommendation: Add the necessary checks to the functions `setBorrowAllowed()`, `setCollateralRatio()`, `setLiquidationIncentive()`, `setLiquidateBorrowAllowed()` and `setRepayBorrowAllowed()` to prevent inconsistent configuration for the protocol.

Update:

1. Fixed.
2. Fixed.
3. Fixed.
4. Fixed.

Fixed in commit 87c2c95.

QSP-9 Unbounded Iteration on Collateral Assets

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`

Description: Many functions within the `BalanceSheetV2` contract call the `BalanceSheetV2.getHypotheticalAccountLiquidity()` in order to determine whether the user's liquidity is healthy. The `BalanceSheetV2.getHypotheticalAccountLiquidity()` function iterates over all collateral and bond tokens held by the vault. The number of bonds a vault holds is limited by an upper bound `maxBonds` defined within the `Fintroller` contract. However, the number of different collateral assets held by the vault is not bounded. The unbounded iteration could lead to any function which relies on the `BalanceSheetV2.getHypotheticalAccountLiquidity()` function running out of gas. This will result in the complete unusability of the vault and the inability to recover any funds stored within it. The same issue exists for the `internal` function `BalanceSheetV2.removeCollateralFromList()`, which may also iterate over every collateral the vault holds.

Recommendation: Implement an upper bound for the number of collateral assets held by a vault.

Update: Mitigated in commit b6a9a63 by adding a new state variable `maxCollaterals` and checking against it in `depositCollateral()`.

QSP-10 Potential Re-Entrancy/Checks-Effects-Interactions Pattern Violations

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`, `packages/proxy-target/contracts/HifiProxyTarget.sol`, `packages/amm/contracts/HifiPool.sol`

Description: As a best practice and to prevent unwanted external contract side effects, it is advised to adhere to the "Checks-Effects-Interactions"-pattern, even in the presence of reentrancy guards.
Following instances were observed where said pattern was violated:

- `HifiProxyTarget.addLiquidity()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.borrowHTokenAndAddLiquidity()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.buyHToken()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.buyHTokenAndAddLiquidity()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.buyHTokenAndRepayBorrow()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.depositCollateral()`: Performs external contract calls (`collateral.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.depositUnderlyingAndMintHTokenAndAddLiquidity()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.

- `HifiProxyTarget.removeLiquidity()`: Performs external contract calls (`hifiPool.underlying().safeTransfer()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.removeLiquidityAndRedeem()`: Performs external contract calls (`hToken.underlying().safeTransfer()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.removeLiquidityAndSellHToken()`: Performs external contract calls (`hifiPool.underlying().safeTransfer()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.sellUnderlyingAndRepayBorrow()`: Performs external contract calls (`underlying.transferFrom()`), before modifying state variables in following sub-calls.
- `HifiProxyTarget.depositUnderlyingInternal()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.
- `HifiPool.sellUnderlying()`: Performs external contract calls (`underlying.safeTransferFrom()`), before modifying state variables in following sub-calls.

Recommendation: Consider re-structuring the code, such that it no longer violates the "Checks-Effects-Interactions"-pattern and/or add reentrancy guards at corresponding calling locations.

Update: From the team :

```
1. HifiProxyTarget is completely stateless (doesn't have any internal state).
2. Every time underlying.safeTransferFrom() or collateral.safeTransferFrom() is called to transfer tokens from the user's EOA to the proxy target, it is in order to use those tokens in a subsequent subcall. It’s not
possible to use those tokens in the proxy before they're transferred to it.
3. The only other contract affected by this other than HifiProxyTarget is HifiPool.
But there is still no internal state being modified after the underlying.safeTransferFrom
interaction.
```

QSP-11 Unlimited Approval in `HifiProxyTarget.approveSpender()`

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: The contract approves an address to transfer tokens on its behalf without setting a limit of how many tokens may be transferred (`token.approve(spender, type(uint256).max);`). If the approved address becomes hacked or is intentionally malicious, it may transfer out all the approved tokens.

Recommendation: We recommend removing unlimited approvals and approve only the amounts that need to be transferred in a given transaction.

Update: From the team:

```
User experience will suffer if they are required to approve every time they interact with the proxy. We believe this is an acceptable trade off.
```

QSP-12 Front Run Can Lead to Loss in Incentive

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/core/fintroller/Fintroller.sol`

Description: Each collateral has its own liquidation incentive, and the owner can change this value using the `setLiquidationIncentive()` function. The issue here is that before running the liquidation, a change request can be made by the owner to front-run the liquidation transaction. The worst-case scenario is that the user will receive less than what was expected.

Recommendation: Consider adding the parameter `liquidationIncentive`, and if it's different from the one stored in the contract, revert.

Update: From the team:

```
Governance will be the owner of this contract, with a waiting period for any changes. Front running a transaction using this method is not a plausible attack.
```

QSP-13 Owner Can Override a Bond

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/core/fintroller/Fintroller.sol`

Description: The owner has the ability to add a new bond token using the `listBond()` function. This will create the new bond with the default parameters. However if the owner injected an existing bond he will override the old bond.

Recommendation: Consider verifying if the bond already existed before inserting it into the `bonds` mapping.

Update: The Hifi team indicated that this is the intended behavior.

QSP-14 Usage of `transfer()` Instead of `safeTransfer()`

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/amm/HifiPool.sol`, `proxy-target\contracts\HifiProxyTarget.sol`

Description: The ERC20 standard token implementation functions also returns the transaction status as a Boolean. It is good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with `require()` to ensure that when the intended ERC20 function call returns `false`, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer didn't.

Recommendation: Use the `safeTransfer` function from the safeERC20 Implementation,.

Update: Fixed in commit ae9459f by replacing the unsafe transfer function with `safeTransferFrom()`, as suggested.

QSP-15 Loss of Precision

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `packages/amm/HifiPool`

Description: The `getBurnOutputs()` function calculated the number of `hTokensReturned` using the following formula

```
hTokenReturned = (poolTokensBurned * hTokenReserves) / supply;
```

If the `poolTokensBurned * hTokenReserves` is less than `supply`, `hTokenReturned` will be equal to `0` due to a loss of precision.

Recommendation: Consider verifying if `poolTokensBurned * hTokenReserves` is greater than `supply` .

Update: From the team:

```
This is not an issue as the code already takes that case into consideration at HifiPool.sol#L231 (https://github.com/hifi-finance/hifi/blob/b9282b6058 6e06937621d8e208295ce56c23cb17/packages/amm/contracts/HifiPool.sol#L231)
```

QSP-16 No Storage Gap Declared on `OwnableUpgradeable`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `packages/protocol/contracts/access/OwnableUpgradeable.sol`

Description: The `OwnableUpgradeable` contract is an `abstract` upgradeable contract. Contracts that inherit from `OwnableUpgradeable`, such as `BalanceSheetV2`, may introduce new storage variables in future versions. To allow for additional storage variables being added to future versions of `OwnableUpgradeable` without introducing storage collisions with `BalanceSheetV2`, a storage gap should be added to `OwnableUpgradeable`.

Recommendation: Add a storage gap such as `uint256[50] private __gap;` at the end of `OwnableUpgradeable`.

Update: Fixed in commit f48f22f by adding state variable `uint256[50] private __gap;` to `OwnableUpgradeable.sol`, as suggested.

QSP-17 User Can Receive a Discount Due to Loss of Precision

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`

Description: Several functions within the protocol lack sufficient validation of their input parameters. We have listed all missing checks below:

Description: The `getQuoteForBuyingHToken()` function is referenced by the `buyHToken()` function to determine how many underlying tokens are needed to buy a certain amount of hTokens. However, the resulting amount of underlying tokens is denormalized: `underlyingIn = denormalize(normalizedUnderlyingIn);`. This results in any amount of underlying token below `underlyingPrecisionScalar` essentially being discounted from the price. The result could be significant if the underlying token has much fewer decimals than `18` and the value of `1` full token is high. For example, in the case of WBTC, a user could receive up to `1` Satoshi worth of a discount. A similar issue exists in `getQuoteForSellingHToken()`, which results in a user paying more hTokens than they should for a certain amount of underlying tokens.

Recommendation: Redesign the `buyHToken()` and `sellHToken()` function so that users cannot receive an incorrect number of tokens due to the loss in precision. Alternatively, remove the `buyHToken()` and `sellHToken()` functions to force users to use the `buyUnderlying()` and `sellUnderlying()` functions, which are not vulnerable to the same loss in precision.

Update: From the team:

```
Underlying tokens will be decided by governance and should never be added if the decimals would result in significant loss of precision.
```

QSP-18 Missing Input Validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`, `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`, `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: Several functions within the protocol lack sufficient validation of their input parameters. We have listed all missing checks below:

- `HifiPool`
  - `constructor()`: Validate that the `string` parameters are not empty.
  - `buyHToken()`: Validate that `_to` is not the zero address.
  - `buyUnderlying()`: Validate that `_to` is not the zero address.
  - `sellHToken()`: Validate that `_to` is not the zero address.
  - `sellUnderlying()`: Validate that `_to` is not the zero address.
- `BalanceSheetV2`
  - `withdrawCollateral()`: Validate that `collateral` is listed within the `Fintroller`.
- `HToken`
  - `constructor`: Validate that the ` string parameters are not empty.
- `ChainlinkOperator`
  - `setFeed()`: Validate that `asset` has `18` or fewer decimals.
- `HifiProxyTarget`
  - `sellHToken()`: Validate that `minUnderlying` is greater than zero.

-
  . `sellHTokenWithSignature()`: Validate that `minUnderlying` is greater than zero.

  . `sellUnderlying()`: Validate that `minHTokenOut` is greater than zero.

  . `sellUnderlyingAndRepayBorrow()`: Validate that `minHTokenOut` is greater than zero.

  . `sellUnderlyingAndRepayBorrowWithSignature()`: Validate that `minHTokenOut` is greater than zero.

  . `sellUnderlyingWithSignature()`: Validate that `minHTokenOut` is greater than zero.

Recommendation: Add the missing validation checks.

Update: The Hifi team believes that none of the missing checks are security issues. We still recommend validating these input parameters to constrain them to valid values and reduce the attack surface.

QSP-19 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`

Description: Many contracts within the protocol feature a contract owner who has special privileges. We recognize that the Hifi team is in the early stages of progressive decentralization. However, all privileged permissions should be clearly documented for users. We have listed all the privileged permissions of the contract owners below:

- `BalanceSheetV2`
  . The `fintroller` contract can be reset anytime, effectively resetting the whole protocol's configuration.

  . The `oracle` contract can be reset anytime, potentially changing the prices used for collateral and debt assets.

- `Fintroller`
  . A new `Bond` can be listed at any time.

  . A new `Collateral` can be listed at any time.

  . The `Collateral` and `Bond` ceilings can be reset anytime, potentially increasing the protocol's risk exposure to a particular asset.

  . A collateral asset's collateral ratio and liquidation incentive can be reset anytime.

  . Collateral deposits can be enabled and disabled at any time.

  . Debt repayments, borrows, and liquidations can be disabled and enabled at any time.

  . The maximum number of bond assets per vault can be reset at any time. Setting this value too high could result in critical functions failing due to excessive gas usage.

- `HToken`
  . The `balanceSheet` contract referenced can be reset at any time.

  . Transfer an arbitrary amount of any token from the contract, except the underlying token.

- `ChainlinkOperator`
  . Price feeds can be added or removed at any time.

Recommendation: All special permissions assigned to the contract owners should be clearly documented in the user-facing documentation. Privileged roles should be secured by large multi-signature wallets and assigned to time-lock contracts to allow users to opt-out of significant changes, such as increased token transfer fees.

Update: From the team:

> Governance will be the contract owner.

QSP-20 Unsafe Casts

Severity: *Low Risk*

Status: Fixed

File(s) affected: `packages/protocol/contracts/oracles/ChainlinkOperator.sol`

Description: The `ChainlinkOperator.getPrice()` function uses an unsafe cast whereby an `int256` is directly cast to a `uint256`: `uint256 price = uint256(intPrice);`. This could result in an overflow if `intPrice` is negative. To resolve this issue, we recommend using the `toUint256()` function from OpenZeppelin's `SafeCast` library.

Recommendation: We recommend using the `toUint256()` function from OpenZeppelin's `SafeCast` library instead of performing a direct cast which could result in an underflow.

Update: Fixed in commit b9282b6 by checking against negative values, as suggested.

QSP-21 Return Values Not Verified

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`, `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: Several contracts within the protocol make external calls to functions with return values which indicate the result of those functions. However, many of these return values are not being validated within the code. We have listed all function calls for which return values are being ignored:
All Calls to HifiPool.buyHToken() All Calls to HifiPool.buyUnderlying() All Calls to HifiPool.sellHToken() All Calls to HifiPool.sellUnderlying()

Recommendation: Validate that the return parameters of external function calls are the expected values.

Update: From the team:

> This is an acceptable risk since HifiPool is an in-house smart contract that is part of the Hifi lending protocol and was developed taking into consideration that its functions would be called externally by EOAs or the Hifi Proxy Target.

QSP-22 Events Emitted After External Calls

Severity: *Informational*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`, `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`

Description: Several instances of events are being emitted after external calls within the contracts of the protocol. If the functions were re-entered due to external calls, this could result in events being emitted out of order. We have provided a non-exhaustive list of such event emissions below:

- `HifiPool`
  - `burn()`: Emits `RemoveLiquidity` after external calls.
  - `buyHToken()`: Emits `Trade` after external calls.
  - `buyUnderlying()`: Emits `Trade` after external calls.
  - `mint()`: Emits `AddLiquidity` after external calls.
  - `sellHToken()`: Emits `Trade` after external calls.
  - `sellUnderlying()`: Emits `Trade` after external calls.
- `BalanceSheet`
  - `borrow()`: Emits `Borrow` after external calls.
  - `depositCollateral()`: Emits `DepositCollateral` after external calls.
  - `withdrawCollateral()`: Emits `WithdrawCollateral` after external calls.
- `HToken`
  - `depositUnderlying()`: Emits `DepositUnderlying` after external calls.
  - `mint()`: Emits `Mint` after external calls.
  - `withdrawUnderlying()`: Emits `WithdrawUnderlying` after external calls.

Recommendation: Emit all events before external calls where possible.

Update: From the team:

> We believe this is not an issue since the only external calls that are made to smart contracts that are not part of the Hifi lending protocol are for token transfers for underlying or collateral tokens, which can be vetted before adding them to the protocol so that the token transfer calls don't cause reentrancy.

QSP-23 Unlocked Pragma

Severity: *Informational*

Status: Mitigated

File(s) affected: `packages/amm/contracts/HifiPool.sol`, `packages/amm/contracts/IHifiPool.sol`, `packages/amm/contracts/HifiPoolRegistry.sol`, `packages/amm/contracts/IHifiPoolRegistry.sol`, `packages/amm/contracts/YieldSpace.sol`, `packages/protocol/contracts/access/IOwnableUpgradeable.sol`, `packages/protocol/contracts/access/OwnableUpgradeable.sol`, `packages/protocol/contracts/core/balance-sheet/SBalanceSheetV2.sol`, `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/balance-sheet/SBalanceSheetV1.sol`, `packages/protocol/contracts/core/balance-sheet/IBalanceSheetV2.sol`, `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/fintroller/IFintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/core/h-token/IHToken.sol`, `packages/protocol/contracts/external/chainlink/IAggregatorV3.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`, `packages/protocol/contracts/oracles/IChainlinkOperator.sol`, `packages/proxy-target/contracts/HifiProxyTarget.sol`, `packages/proxy-target/contracts/IHifiProxyTarget.sol`, `packages/proxy-target/contracts/external/WethInterface.sol`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity >=0.8.4`. The `>=` before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, we recommend to remove the caret to lock the file onto a specific Solidity version.

Update: The Hifi team mitigated the issue by limiting the range for the floating Solidity version to remain within `0.8.x` to avoid upgrading to a breaking version. However, note that different versions of the compiler within the `0.8.x` range may still be used.

QSP-24 Ownership Can Be Renounced

Severity: *Informational*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

Recommendation: Double check if this is the intended behavior.

Update: From the team:

> This is intended behavior and is not an issue.

QSP-25 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/access/OwnableUpgradeable.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability, or may include intentionally or unintentionally modified upstream libraries. The `OwnableUpgradeable` contract is a fork of an OpenZeppelin contract. We identified a separate vulnerability in the report due to not using the latest OpenZeppelin contract. That vulnerability was the lack of inclusion of a storage gap which is included in the latest OpenZeppelin contracts.

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use a package manager (e.g., npm) for handling library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries. If the file is cloned anyway, a comment including the repository, commit hash of the version cloned, and the summary of modifications (if any) should be added. This helps to improve traceability of the file.

Update: From the team:

> This is a risk we are willing to accept.

QSP-26 Test Code Mixed with Production Code

Severity: *Informational*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/oracles/SimplePriceFeed.sol`, `packages/protocol/contracts/oracles/StablecoinPriceFeed.sol`

Description: To prevent confusion and the introduction of vulnerabilities, test code should not be present in production code/should be clearly separated.

Recommendation: Remove any test-related functionality from production code or move it into dedicated test/mock folders.

Update: From the team:

> This is a risk we are willing to accept.

QSP-27 Transaction Ordering Dependence for `BalanceSheetV2.initialize()`

Severity: *Informational*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`

Description: The various `initialize()` functions of upgradeable contracts are not constructors. There is a low-but-not-zero chance that someone can call these after the contracts have been deployed, but before the development team calls them. Consequently, contracts may get initialized with values that are not desirable by the development team.

Recommendation: Be aware of this issue, and be prepared to redeploy your contracts if these calls are front-run. Do not use your project until you have checked that your calls to these functions went through.

Update: From the team:

> This is a risk we are willing to accept. We would not go public with any upgradable smart contracts before they're first initialized.

QSP-28 Minimal/incomplete Pause Mechanism

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/amm/contracts/HifiPool.sol`

Description: The Hifi protocol allows for pausing numerous operations for collateral and debt assets through various functions within the `Fintroller` contract. However, many other critical operations are not pausable. While we realize that there is always a trade-off between centralization and stronger pause functionality, the Hifi team should consider adding the ability to pause the following:

- Bond/debt operations
  . Withdraw underlying deposited within the `HToken` contract. This would prevent an attacker from withdrawing maliciously obtained hTokens.
- Collateral operations
  . Withdrawal of collateral stored within the `BalanceSheetV2` contract. This could prevent an attacker from stealing or withdrawing excess collateral.
  . Liquidation of collateral assets stored within the `BalanceSheeetV2` contract. While the ability to pause operations on particular bond assets has been added, there is no way to disable the liquidation of a particular collateral asset. This may be desirable if the oracle temporarily delivers incorrect prices for a specific asset.
  . Prevent collateral from counting towards the user's total collateral value. If the team deems a collateral asset too dangerous to continue using, they may consider not allowing it to count towards a user's collateral value. Another mitigation would be raising the collateral ratio to the maximum value.
- Trading operations
  . No pause functionality has been added for the AMM portion of the protocol. The team could consider adding functionality to pause adding/removing liquidity and trading functions.

Recommendation: Consider implementing the additional suggested pause functionality.

Update: From the team:

> This is a risk we are willing to accept.

QSP-29 Reliance on Unaudited Contracts and Libraries

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `packages/amm/contracts/HifiPool.sol`, `packages/amm/contracts/HifiPoolRegistry.sol`, `packages/amm/contracts/YieldSpace.sol`, `packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol`, `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/h-token/HToken.sol`, `packages/protocol/contracts/oracles/ChainlinkOperator.sol`, `packages/proxy-target/contracts/HifiProxyTarget.sol`

Description: Numerous contracts within the contract either inherit from PRB contracts or use PRB libraries. These contracts are not within the scope of the audit and have not been previously audited as stated by the author. These contracts might have existing unfound vulnerabilities which could compromise the security of the Hifi protocol.

Recommendation: Have the PRB contracts audited, or swap them for audited OpenZeppelin contracts.

Update: From the team:

> This is a risk we are willing to accept.

QSP-30 `ChainlinkOperator` Does Not Allow for Multiple Assets with the Same Symbol

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/oracles/ChainlinkOperator.sol`

Description: Function `ChainlinkOperator.setFeed()` saves new feed information in storage variable `mapping(string => Feed) internal feeds;`, which is indexed by the assets symbol string. Trying to add different assets with the same symbol string would lead to collisions in said array, overwriting the existing asset.

Recommendation: Clarify if this is intentional/problematic in the planned business logic and planned tokens or consider changing the array index to be a hash of the symbol string and asset address, to be able to disambiguate between multiple assets with the same symbol string.

Update: From the team:

> This is intended behavior and could easily be avoided by never adding support for 2 different tokens to the protocol that have the same token symbol.

QSP-31 Protocol only Works with Tokens with up to 18 Decimals

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `packages/protocol/contracts/core/fintroller/Fintroller.sol`, `packages/protocol/contracts/core/fintroller/Fintroller.sol - ./packages/protocol/contracts/core/h-token/HToken.sol`

Description: The protocol currently only allows adding tokens as collateral or underlying for HTokens with up to 18 decimals and not more. While in practice most tokens only use up to 18 decimals, this constraint could become a limitation in the future, especially given that this value is hardcoded and cannot be easily modified.

Recommendation: Clarify if this is intentional and list this limitation in public-facing user documentation or consider making this decimal requirement a changeable variable, rather than a hardcoded value.

Update: Fromt the team:

> This is intended behavior. Any tokens listed in the project would be vetted to have no more than 18 decimals.

Code Documentation

1. `HifiPool` or `IHifiPool` should include code comments to indicate that the functions should only be called from a contract that makes safety checks, such as ensuring acceptable slippage.
2. Some events, such as `TrackPool` and `UntrackPool` in `IHifiPoolRegistry`, are not documented using the NatSpec standard.
3. The `YieldSpace.hTokenOutForUnderlyingIn()` and `YieldSpace.underlyingOutForHTokenIn()` functions both have a code comment indicating the formula: `y = (x_s^(1-gt) + y_s^(1-gt) - x^(1-gt))^(1/(1-gt))`. However, both these functions solve for `x` rather than `y`. It would improve code readability if the equation was expressed in the from `x = ...`;
4. The code comment above `IBalanceSheetV2.getHypotheticalAccountLiquidity()` states `@param collateralAmountModify The hypothetical normalized amount of collateral`. However, `collateralAmountModify` is the unnormalized amount of collateral.
5. The code comment above `IFintroller.listCollateral()` states, `It is not an error to list a bond twice`. However, it should state, ' It is not an error to list a collateral asset twice` .
6. The following typographical errors have been noted:
    1. `IFintroller.sol#L99`: `deposit` -> `depositing`.
    2. `IChainlinkOperator.sol#L58`: `it have` -> `it to have`.
    3. `IHToken.sol#L176`: `Throws` -> `Emits`.
    4. `IHifiPool.sol#L216`: `investigated` -> `invested`.
    5. `IHifiPool.sol#L237`: `underlying` -> `hToken`.
    6. `IHifiProxyTarget.sol#L50`, `L64` and `L95`: `underlyingAmount` -> `underlyingOffered`.
    7. `IHifiProxyTarget.sol#L295`: `maxHTokenAmount maxHTokenAmount` -> `maxHTokenAmount`.
    8. `IHifiProxyTarget.sol#L329`, `L343`, `L361`, `L382`, `L409`, `L429`: `collateralAmount` -> `depositAmount`.

9. `YieldSpace.sol#L19`: `out` -> `in`.

7. Missing or incorrect NatSpec comments:
    1. `IFintroller.SetCollateralRatio()`: Incorrect NatSpec comment for parameter `collateral` (Parameter is `The related collateral`, not `The related HToken`).
    2. `IFintroller.SetDepositCollateralAllowed()`: Missing NatSpec comment for parameter `collateral`.
    3. `IFintroller.SetMaxBonds()`: Incorrect NatSpec comments for parameters `oldMaxBonds` and `newMaxBonds` (old/new max bond values are of type `uint256` and not `address`).
    4. `Fintroller.isCollateralListed()`: Duplicate NatSpec comment from `IFintroller.sol` (Consider using `@inheritdoc IFintroller`).
    5. `Fintroller.setDepositCollateralAllowed()`: Missing NatSpec comment mentioning constraint `The collateral must be listed`.
    6. `Fintroller.setDepositUnderlyingAllowed()`: Missing NatSpec comment mentioning constraint `The bond must be listed`.
    7. `IHToken.depositUnderlying()`: Missing NatSpec comment mentioning that function also emits a `Transfer()` event.
    8. `IHToken.redeem()`: Missing NatSpec comment mentioning that function also emits a `Transfer()` event.
    9. `HToken.normalize()`: Wrong NatSpec keyword for return value `normalizedUnderlyingAmount` (`@return` not `@param`).
    10. `IBalanceSheetV2.getBondList()`: Missing NatSpec comment for return value.
    11. `IBalanceSheetV2.getCollateralAmount()`: Missing NatSpec comment for return value.
    12. `IBalanceSheetV2.getCollateralList()`: Missing NatSpec comment for return value.
    13. `IBalanceSheetV2.getDebtAmount()`: Missing NatSpec comment for return value.
    14. `IBalanceSheetV2.liquidateBorrow()`: Order of NatSpec comments for parameters `borrower` and `bond` are switched.
    15. `IHifiPool.HifiPool__BurnZero()`: Incorrect NatSpec comment (copy-and-paste from `HifiPool__MintZero()`).
    16. `YieldSpace.underlyingInForHTokenOut()`: Incorrect NatSpec comment for parameter `hTokenOut` (copy-and-paste from `hTokenOutForUnderlyingIn()`).
    17. `HifiPool.denormalize()`: Incorrect NatSpec keyword for return value `underlyingAmount` (`@param` -> `@return`).
    18. `HifiPool.normalize()`: Incorrect NatSpec keyword for return value `normalizedUnderlyingAmount` (`@param` -> `@return`).
    19. `HifiProxyTarget.denormalize()`: Incorrect NatSpec keyword for return value `denormalizedAmount` (`@param` -> `@return`).
    20. `HifiProxyTarget.normalize()`: Incorrect NatSpec keyword for return value `normalizedAmount` (`@param` -> `@return`).

8. Unclear/Unrelated comment in `HifiProxyTarget.sol#L244` (`"normalizedUnderlyingRequired" was denormalized to "underlyingRequired", offsetting the trailing 12 digits.`).

Adherence to Best Practices

1. The following storage variables should be marked as immutable:
    1. `HifiPool.hToken`
    2. `HifiPool.maturity`
    3. `HifiPool.underlying`
    4. `HifiPool.underlyingPrecisionScalar`
    5. `HToken.fintroller`
    6. `HToken.maturity`
    7. `HToken.underlying`
    8. `HToken.underlyingPrecisionScalar`

2. The following `external` or `public` functions should not be prefixed with `_`. Functions prefixed with `_` are usually `internal` functions.
    1. `OwnableUpgradeable._renounceOwnership()`
    2. `OwnableUpgradeable._transferOwnership()`
    3. `HToken._setBalanceSheet()`

3. Rename the `hTokenOut` parameter of `HifiProxyTarget.getUnderlyingRequired()` to `hTokenToProvide` or an alternative which indicates what it represents within the context of the function.
4. To improve the gas usage of all for-loops within `BalanceSheetV2`, make the following modifications:
    1. Use an `unchecked` block to increment the counter.
    2. Do not explicitly assign `i` a value. It will be zero by default.
    3. Use `++i` instead of `i++`.

5. The `BalanceSheetV2.borrow()` function can be made `external` to save gas.
6. Initialize the properties of structs separately to save gas. This reduces gas usage by not explicitly initializing some properties to their defaults.
7. To ensure proper current and future compliance, all contract files should be marked with a license. In this regard, consider adding a license to contract `WethInterface.sol`.
8. For readability and consistency, code should adhere to the commonly accepted code style guide. In this regard, consider the following instances, deviating from said guide:
    1. Constants `ChainlinkOperator.pricePrecision` and `ChainlinkOperator.pricePrecisionScalar` are not adhering to be named in all capital letters with underscores separating words.

9. For improved readability and code quality it is advised to remove duplicate or unused code. In this regard consider the following cases:
    1. `HToken.sol#L210-214`: Performs an explicit underflow check for to-be-withdrawn underlying amount against the depositors balance. However, as a solidity version >0.8.0 is used (`pragma solidity >=0.8.4;`), solidity performs implicit arithmetic underflow & overflow checks, making said check obsolete. Clarify if this is kept in order to be able to revert with `HToken__WithdrawUnderlyingUnderflow()` or, if not, consider removing this snippet to save gas.

10. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the (other) address parameters in
    1. `IBalanceSheetV2.SetFintroller()`,
    2. `IBalanceSheetV2.SetOracle()`.

11. To improve readability and maintainability, it is recommended to use descriptive code comments and meaningful names when naming variables, functions, .. . In this regard, consider renaming following instances:
    1. To better disambiguate the code comments between `IBalanceSheetV2.repayBorrow()` and `IBalanceSheetV2.repayBorrowBehalf()`, consider changing `Erases the borrower's debt` to `Erases the callers debt` in `IBalanceSheetV2.sol#L294`.

12. To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. Following instances should therefore be changed accordingly:
    1. `HToken.sol#L66`, `L84`, `L87`, `Fintroller.sol#L164`, `BalanceSheetV2.sol#L122`, `L187`, `L229` and `HifiPool.sol#L51`: `18`.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
93dd0e6e63c5391f579d5e651c7f2b77ac16d5866a42b97cf175fcf23abc2c20 ./packages/proxy-target/contracts/HifiProxyTarget.sol
268637cd7cb1911084146bb2c7664c274ffcc7c9822d5b7ec56240dd6ad8e4bc ./packages/proxy-target/contracts/IHifiProxyTarget.sol
58650d59c7eda3e8783a32d3bcc4f464dcaacca19000dc633e78eef4ee33b2ae ./packages/proxy-target/contracts/external/WethInterface.sol
7d4cba342aaca15667b262085d9449c0036ae1884667a8f08c73d6c23e1d50ae ./packages/protocol/contracts/test/BalanceSheetUpgraded.sol
b111b9a2045cf9d9663b882bed17a7f755439226563bcbf2752060563faba344 ./packages/protocol/contracts/test/GodModeBalanceSheet.sol
e433296427e558591ef434237be517bb9a0dd5a4426d8ba42f6dbcf80b299b9b ./packages/protocol/contracts/test/GodModeErc20.sol
bbe424b3881278e3b1f26362d60fcc07507fdf19fc0b1b30da5495dfacb8f856 ./packages/protocol/contracts/test/GodModeHToken.sol
a16304288ecf6e2c27a9a7eefce8dc6a234134312ea2e0f62eb9beca7e76d870 ./packages/protocol/contracts/test/GodModeOwnableUpgradeable.sol
194e2c6e954ec6b9b3fed6144433377946e2f89871819240727a94c13d57f2cf ./packages/protocol/contracts/oracles/ChainlinkOperator.sol
53207dc18eb9f78eb74b3ec3a8b56f6b2a867377462d8d54434f0a70f0c0eaa9 ./packages/protocol/contracts/oracles/IChainlinkOperator.sol
5364b099684bc40b350905e5a5b1ed4c181b776b62d9cff7a7d3444fdb0f9e6c ./packages/protocol/contracts/oracles/SimplePriceFeed.sol
8144012fa7e729cd3370921b62eac081dd113625fae76522a8a245affdfa2a75 ./packages/protocol/contracts/oracles/StablecoinPriceFeed.sol
f0675b95c18654dbfe1455e2ba7cbdecbbb4dec78461eb20baf1e2caec255ca6 ./packages/protocol/contracts/external/chainlink/IAggregatorV3.sol
74053291581dadcee25417c2d4d1abaf88b11c84d11f43364761fbd7792f2d76 ./packages/protocol/contracts/core/h-token/HToken.sol
e083d53edd48e38b70d319c2e85afbbaf37ba6aade6eb04a27a2fe0a2527ff89 ./packages/protocol/contracts/core/h-token/IHToken.sol
6d3598185955be78d8b2b558169c0b4b7ca4a1a04a6773798cd84166378fdcc6 ./packages/protocol/contracts/core/fintroller/Fintroller.sol
82ca378c6395b1698846cb70dfff07d79c91388bdd71dae61a966cb2b89ab4f5 ./packages/protocol/contracts/core/fintroller/IFintroller.sol
0e66d02c86f7fd1d5228c719314f1f324862ecb810707c6941865ad9a2de6e30 ./packages/protocol/contracts/core/balance-sheet/BalanceSheetV2.sol
f702c99703579431faa5a74434d74c9533c6c9270e64268005e80beec1832274 ./packages/protocol/contracts/core/balance-sheet/IBalanceSheetV2.sol
de5009d3c5db9d2cceb050a62ff46c794b4120b2f486dcac5041756eff94dc5e ./packages/protocol/contracts/core/balance-sheet/SBalanceSheetV1.sol
1d4d6349880f04f479e055df6d4fdd7de4aec94703b2ab9cdacabc9ca8feac45 ./packages/protocol/contracts/core/balance-sheet/SBalanceSheetV2.sol
a774ea75c914a1a14f6c4ba806ab68f5eb2d6b842b7ff7b0f4651628da9e1c93 ./packages/protocol/contracts/access/IOwnableUpgradeable.sol
0186015528e41026b6e8e59a5a2434eae5d4e87f2efa0e7ca19767762d2feb36 ./packages/protocol/contracts/access/OwnableUpgradeable.sol
d067537816fa77ff2694ae756329fcfe5db1dff8f6e3fc02dcd12167e0d72d97 ./packages/amm/contracts/HifiPool.sol
cfec738aa4e592a226646b7a60774acda82c1fcdd05627dc360d51111eccf5fb ./packages/amm/contracts/HifiPoolRegistry.sol
ebbd4e12235c4ce53018c1fc215a12df77a5cba150ae5baa153be323e9da7d29 ./packages/amm/contracts/IHifiPool.sol
b1583ca6363250bcd32cb1aaec49a52b6437399f9cfee7bf137daf0a2bacdc6e ./packages/amm/contracts/IHifiPoolRegistry.sol
e433296427e558591ef434237be517bb9a0dd5a4426d8ba42f6dbcf80b299b9b ./packages/amm/contracts/test/GodModeErc20.sol
7243353ccde811c8672d05e2611141a4e004c98b4e94504009a4aac90f0bb5a7 ./packages/amm/contracts/test/GodModeHifiPool.sol
2b150ca170d5d419933c133c6361068b3ee7f763d34e494c67c40ea56785a82b ./packages/amm/contracts/test/GodModeHifiPoolRegistry.sol
15abf63e6e1fbdff7a8515c88dd1b54848d637b1a9faff6b789fa5f28bfb195e ./packages/amm/contracts/test/GodModeHToken.sol
bce0e663adde0f01fef5f1ba828de1007c455ce02a0a04cd28cc5aadcc6180ca ./packages/amm/contracts/test/YieldSpaceMock.sol
bc4d5e5b1c8b8cd2a576ddf1318523aebbb67097d54998051d54159284e30050 ./packages/amm/contracts/math/YieldSpace.sol
```

Tests

```
d731155b62d3779d07c4a67a2b1994be450fc4e8c76772546c137ae0edc8c061 ./tests/protocol/test/unit/index.ts
c28321b88acdaf2978d9ad5d7d151561d33c3cf5a038e423aa83e65df03e53e0 ./tests/protocol/test/unit/ownable-upgradeable/OwnableUpgradeable.behavior.ts
7ba683ace8600620596caa6ff118337bdb9d8210ef788bcb4395f533d6eb40c3 ./tests/protocol/test/unit/ownable-upgradeable/OwnableUpgradeable.ts
38d9709a42a6a2e146432f3f0aed714eb35e926a609d014df4e2a9acf63be76a ./tests/protocol/test/unit/ownable-upgradeable/view/owner.ts
315ec4b8fdf802667ee236527193fe20a5f815d23aa1138237acec81c6c5d971 ./tests/protocol/test/unit/ownable-upgradeable/effects/renounceOwnership.ts
```

56d5d60878f435aa1116f8675c05e796eed5e60bb346ddc01d8665395d45cc4f ./tests/protocol/test/unit/ownable-upgradeable/effects/transferOwnership.ts

05c47e9a57af9b320e4b016d6f09d7d568d5da1d31946ed7eac277882910e4cc ./tests/protocol/test/unit/h-token/HToken.behavior.ts

c2b3e95fc425370a9c8fef8b9507c980189d0d7d5d55a24bb133998ffc367a11 ./tests/protocol/test/unit/h-token/HToken.ts

41fb49c8c9b56c2b0068bfda9efccc2544ee7ea24cf73885115672fc4d39db71 ./tests/protocol/test/unit/h-token/view/balanceSheet.ts

269d652cc177913320f4df691aeb26450720bcc0e565d6debdc037c57bcb3c01 ./tests/protocol/test/unit/h-token/view/fintroller.ts

ec070adc5b3c686b55e1d5cdff7311c3290ffbdf95b806dc1b9316164d8a8d8b ./tests/protocol/test/unit/h-token/view/isMatured.ts

53001b97a701d09cd5e65bdd4af4e9064f23c61d45f4edd2ada5ea781d8109a7 ./tests/protocol/test/unit/h-token/view/maturity.ts

62039021db0dc80621ea5da088528938c914616ed7629c6e6f554c461d39d330 ./tests/protocol/test/unit/h-token/view/nonRecoverableTokens.ts

16bd8abb6c3c2ac4ad6d07d0365054e7d83d9921d62edc5ba8d0286ed26d51be ./tests/protocol/test/unit/h-token/view/totalUnderlyingReserve.ts

3825c9157da7c5f483c743823816986ec378f3b275511ebc8f2d1c1ca4c80bc9 ./tests/protocol/test/unit/h-token/view/underlying.ts

71805801e6c3f94e63e12f78a48590774b1d6a01a1808710a672e9d8a491040d ./tests/protocol/test/unit/h-token/view/underlyingPrecisionScalar.ts

b40cbb9f9250fbb4cf106eeae59edf4374dfae4961cfa27406fd58fd901472c4 ./tests/protocol/test/unit/h-token/effects/burn.ts

91937dbd137db9cff1750e8cc5cd71b4106cc56213b43c049b1003dbe1134af2 ./tests/protocol/test/unit/h-token/effects/depositUnderlying.ts

0f9ec2814aef8a5aef0971fd58b24b5f5efb3bb05d8164d27a0e83f73f6eda98 ./tests/protocol/test/unit/h-token/effects/mint.ts

28933d2fc6cfd1f41f1a9623e0a18f35553f7d8a204864c6807cf123b6d0e348 ./tests/protocol/test/unit/h-token/effects/redeem.ts

d5037516134ff78a1bf50377f862ab64761829bf6ba273cd55e6d02a53a743d1 ./tests/protocol/test/unit/h-token/effects/setBalanceSheet.ts

a450ed0c986a51b64f0efbeaeca8be4eef255fe6984fcae3f2e89d5162c27b9c ./tests/protocol/test/unit/h-token/effects/withdrawUnderlying.ts

019cadc438463bda276fc7fd3f1912ff5a2d6f7f6817a296c70cd066a868fab3 ./tests/protocol/test/unit/h-token/deployment/constructor.ts

6be652f1647e15256b1e17c350c35fdd4f83f7fce3cd152c0c670d7c4a765ac2 ./tests/protocol/test/unit/fintroller/Fintroller.behavior.ts

9f94b51160e4e69f4e183f2459d9fbce1e15614c62bff969c32fadb699643e8a ./tests/protocol/test/unit/fintroller/Fintroller.ts

45e6e8f5dc992207e912198a027d8107c164d649ae1b131bcfa8beef823d0ff0 ./tests/protocol/test/unit/fintroller/view/getBond.ts

b513442c104424fca09cd288676161dafa04731e791f639bf99392cf1a33c03c ./tests/protocol/test/unit/fintroller/view/getBorrowAllowed.ts

02158888b27d63930c9b2d083c4d99c74c066733fd56331b82418455eabac62f ./tests/protocol/test/unit/fintroller/view/getCollateral.ts

327f68e257f14940780129e2f20e7328325e1983489d7054f86c4a69bcf555c8 ./tests/protocol/test/unit/fintroller/view/getCollateralCeiling.ts

d1b35010cdeea1208cf2dd268beec6fc08eeaf492190a798d18aaeab7c1ce6eb ./tests/protocol/test/unit/fintroller/view/getCollateralRatio.ts

005eccc4a8ecfaa0e166f2d8075ff35cffdaac339a6f3a70801e6d880621a2a8 ./tests/protocol/test/unit/fintroller/view/getDebtCeiling.ts

9c8e0df95636fbe0ce98f553028c8035f771fcde7d0b4a744465f328cd05f119 ./tests/protocol/test/unit/fintroller/view/getDepositCollateralAllowed.ts

67466fa03554b17d4b721902f995b98307019eac3c913ab9a77e8ad0a929aae7 ./tests/protocol/test/unit/fintroller/view/getDepositUnderlyingAllowed.ts

61a26e2e72dcf1e43ed75177814d3853b8273ab7fc546f1b8268eff1f7bc552b ./tests/protocol/test/unit/fintroller/view/getLiquidateBorrowAllowed.ts

4c396a068ea36349b55e4ce473e8b623f3eb4998037c897dcb5c3c459340abe2 ./tests/protocol/test/unit/fintroller/view/getLiquidationIncentive.ts

f7e365f371a343ee6b14eb3c18b9e29dbbfe9a614112301515b1bde209574fa1 ./tests/protocol/test/unit/fintroller/view/getRepayBorrowAllowed.ts

1f25620e4e8cff5d24449e94679e6bc40fa2fe941421d1b66dc54600c1a9b139 ./tests/protocol/test/unit/fintroller/view/isBondListed.ts

242238e2593a8d311d81055ae8531f1181c5cf1bb29614ce4c793a2a95633f03 ./tests/protocol/test/unit/fintroller/view/isCollateralListed.ts

2bcd0d896e7fd88556b59712f02c472b11006b5a53f8eb7d6554a9a45d070bf7 ./tests/protocol/test/unit/fintroller/view/maxBonds.ts

1a7075a38f0dbeecea6a2706e2c7e0f1a802eac066c8bc9ea8ba21458af59a7f ./tests/protocol/test/unit/fintroller/effects/listBond.ts

f94a98dc3be79f04bfb7bab17fc80c3361f63aaf67db1c6c8403b0a12788d3e4 ./tests/protocol/test/unit/fintroller/effects/listCollateral.ts

6be48c4390efff54168c874176dbf838f9bc80ab512b2265694f0b3fa6569f3e ./tests/protocol/test/unit/fintroller/effects/setBorrowAllowed.ts

ba499aeca95e263176e9b6692e5fd34d039aabb5ec06cd973b09f52dea458ae0 ./tests/protocol/test/unit/fintroller/effects/setCollateralCeiling.ts

68e4c2c4bc78b0b41a343231c6c5809e0699e16b11d6ecb8ac0d5685da63c640 ./tests/protocol/test/unit/fintroller/effects/setCollateralRatio.ts

ae56514fb8f9076288e4a9c1ee6d4067d1e28782632edc000ae6f1113a2a0f4e ./tests/protocol/test/unit/fintroller/effects/setDebtCeiling.ts

24799729486168d04af30e67d57c7ef31c6ebe1126d9091869e2cb440252e79c ./tests/protocol/test/unit/fintroller/effects/setDepositCollateralAllowed.ts

de853527382a8911db7a5226dafb5ef366b578b9a12fb2541cb00a88310a648c ./tests/protocol/test/unit/fintroller/effects/setDepositUnderlyingAllowed.ts

7cadcc10dc666f575569e8a681b7ef41bb6b1dd900ad0a7f0f3e34e6bb6d5e7f ./tests/protocol/test/unit/fintroller/effects/setLiquidateBorrowAllowed.ts

f8d81b0499c99496faa8a66dc9b9f6d79ec5af823a6366efbb4c854c171c8113 ./tests/protocol/test/unit/fintroller/effects/setLiquidationIncentive.ts

6d3368f7c9addc6c1f16292a6a5637813c79c22646d588a52a377e82145e659b ./tests/protocol/test/unit/fintroller/effects/setMaxBonds.ts

ad54e66ab52cab6cf8f2c5a8d6a58cf8475a9d5e6acc2ddd92a798d2e9a8ea6f ./tests/protocol/test/unit/fintroller/effects/setRepayBorrowAllowed.ts

089299e96d5d58c099d7fb1b1531e2a95f8a5c9439280565670c8c016ed571e8 ./tests/protocol/test/unit/chainlink-operator/ChainlinkOperator.behavior.ts

801caa0b1445faf75846695b18fded86b36463a1e3db17dde23693810d4f25fc ./tests/protocol/test/unit/chainlink-operator/ChainlinkOperator.ts

4f53260b34faa724eb10f485304b3bceebd7fbea73a2bc9fcfd9baa2c65a1840 ./tests/protocol/test/unit/chainlink-operator/view/getFeed.ts

e8f73b4071e1335be84c30022b1253672b3c85bfc48498a5f1926b7e78f9012d ./tests/protocol/test/unit/chainlink-operator/view/getNormalizedPrice.ts

6ac77cea81baa21176f7487c500233792862ecd988b52cbcba4f31ef0a554354 ./tests/protocol/test/unit/chainlink-operator/view/getPrice.ts

5b903b5f1ee1a1a5fa27c785c7bb716c575fdfe8115e9dbb5f148a75fb95db90 ./tests/protocol/test/unit/chainlink-operator/effects/deleteFeed.ts

88b6e60e2b102ae2331e92231df9fa6f3e812909ee82b38734071be0f21f0d97 ./tests/protocol/test/unit/chainlink-operator/effects/setFeed.ts

c81a45806d6b41aaef2c68583582eb80438a0f0fe24071645a87ddb2e3d09a71 ./tests/protocol/test/unit/balance-sheet/BalanceSheet.behavior.ts

5d87a32b325995810d10451dcbe4b4136fb20d1a3b603a211f09d197cdbecd28 ./tests/protocol/test/unit/balance-sheet/BalanceSheet.ts

93466188e3d41d301affec8db2315bbc4497c66d6b014686034885e281c7e507 ./tests/protocol/test/unit/balance-sheet/view/fintroller.ts

8436f2f4c44e1083301642ea0a45fb1da2b4c93ca0698ebdb26fcdacaeb8e134 ./tests/protocol/test/unit/balance-sheet/view/getBondList.ts

613cec935a6d2ff70fd63d4ac01eddfe66ff59a049de192be2b7c35910a6b278 ./tests/protocol/test/unit/balance-sheet/view/getCollateralAmount.ts

087b2758cfbefe80558df63ceebff944041b79a6a0138d9a270ddec815e3baf5 ./tests/protocol/test/unit/balance-sheet/view/getCollateralList.ts

782c427d4c75b35308ec9dd269e24f503b76b31dd6fe65c28c31b1f775be1c18 ./tests/protocol/test/unit/balance-sheet/view/getCurrentAccountLiquidity.ts

4c8db3471e527a61e10b5430f3f3815f4013bef7c280574a1254575366931483 ./tests/protocol/test/unit/balance-sheet/view/getDebtAmount.ts

cb50f33186462c32df8b8e82c51fc0677738f809404917785fc909b8c854ba55 ./tests/protocol/test/unit/balance-sheet/view/getHypotheticalAccountLiquidity.ts

2c4986b2ad1cdaef63a3de3f5bed2913272e5f07ef1d698f2975e9ec36006291 ./tests/protocol/test/unit/balance-sheet/view/getRepayAmount.ts

d2df64b256f0312f1d1fa2ddf1c79ee2071d2ead06b80b724de99c32661b202d ./tests/protocol/test/unit/balance-sheet/view/getSeizableCollateralAmount.ts

5d2ad0d550c87dae33deafe372cb099916303c2f4004adce1fdb22cec7e23c1f ./tests/protocol/test/unit/balance-sheet/view/oracle.ts

01aea83efde80e5440aca3a546f8b23892806ec8a1db2057bb69c4d1c7ea0dbd ./tests/protocol/test/unit/balance-sheet/effects/borrow.ts

2f89d6d3420f20de080856f89674345347b0a2723bc20f440093335e229b3b5b ./tests/protocol/test/unit/balance-sheet/effects/depositCollateral.ts

578262a12d7e562bbf7b84357a974b332e719ad987f39d2e2c4b9981406fbee2 ./tests/protocol/test/unit/balance-sheet/effects/liquidateBorrow.ts

c3eba00246ebbb7810b28a96aebd4c4c3543f06051d4c4fa4850dbec68602e1f ./tests/protocol/test/unit/balance-sheet/effects/repayBorrow.ts

8d18e856cf666f7836599e820c0f3d32f7433333df4d44a3f17ccea1c68163bb ./tests/protocol/test/unit/balance-sheet/effects/repayBorrowBehalf.ts

51faa6ca80d76a84a9758354a960ae3a72f0dc28de4d5001c0755f4c0dc42e2c ./tests/protocol/test/unit/balance-sheet/effects/setFintroller.ts

ef47ff968a92ccd2556c36062c13adf3fdba94b0b3303b26f0053aab86c2bbcb ./tests/protocol/test/unit/balance-sheet/effects/setOracle.ts

b0a0c4b8b2047d4e10ab5fa83233353097093006e9a22b5ab716f5c3d4b90774 ./tests/protocol/test/unit/balance-sheet/effects/withdrawCollateral.ts

c6693d10b703346831fc13962643e58a76e02c65249c1b6054450ca35b777441 ./tests/protocol/test/unit/balance-sheet/deployment/proxy.ts

a11f95e25c9755dc4cd939e8ece8f7b14cb38710c8dffbe017a88d8e07ed83f5 ./tests/protocol/test/shared/contexts.ts

e36d293ef19ae316d37cd3f8dac40e83c29cf73e45657fe0f6e223ce0d09b1af ./tests/protocol/test/shared/deployers.ts

15038abc68dc44d422876fcd914eff6129eac24c9a6b8f94c86e106760a05f4b ./tests/protocol/test/shared/fixtures.ts

fb88121e2ef18cab2516c7286e0e53db43ca8d8e54d2789ace8ffaa1c8b67f9b ./tests/protocol/test/shared/mirrors.ts

372dcc915d0f928b7e55f5552f7e0150c57cd491505615af4f99e85b6172e5ce ./tests/protocol/test/shared/mocks.ts

7f21bf081803d76668154e8dd5d243250e7a48cc4bf53142ff31b40023c0ece2 ./tests/protocol/test/shared/types.ts

0804baa65aa1383d5de3eb372ba002124d349fa778c33aeb9723518f22de63e5 ./tests/protocol/test/integration/index.ts

1732bc52d0ffa69178b80a867ac89b53af63f7800de83b7831d3ca5af027ddb6 ./tests/protocol/test/integration/balance-sheet/BalanceSheet.behavior.ts

333c28531f436746ff24de38acc4efec5260213eed430623ded1eadf78d59d45 ./tests/protocol/test/integration/balance-sheet/BalanceSheet.ts

0249c905755e0df4badf8f28304a6fd8ee537c37438eafc9afd29da449679b98 ./tests/protocol/test/integration/balance-sheet/effects/borrow.ts

6465829f10a6643725c3b6a3c5f88ce4cb1fae7a07805f4d8c83cca6b6992be4 ./tests/protocol/test/integration/balance-sheet/effects/liquidateBorrow.ts

1f404d1078034a1cf5c049406cd8603b170a7a53bd4ebdec9a9bdd3e55b013fc ./tests/protocol/test/integration/balance-sheet/effects/repayBorrow.ts

40fd4aeb4adcc1712ee6107d3beafc633cc92f42eec5aac2716d95b43772edfe ./tests/amm/test/unit/index.ts

5a4cc5f42ae0090cab41159e00740237dee2908052caebc806b2618444be5b86 ./tests/amm/test/unit/yieldSpace/YieldSpace.behavior.ts

14cd3fa94f6dc2a1d25792bc51608ad5128ee9cc05a47f449cf8f956ac899119 ./tests/amm/test/unit/yieldSpace/YieldSpace.ts

c86ab9b06fc66e4a4cc7f5b323d90486d4848981121d1afbe891f3821d1dc16f ./tests/amm/test/unit/yieldSpace/pure/getYieldExponent.ts

371f6e230f816ee67fdcdcd86ed9ea5c3cca9f1a05ddadec54c40471070cdb16 ./tests/amm/test/unit/yieldSpace/pure/hTokenInForUnderlyingOut.ts

d6ebefcde33535713aba8a6d7da20569fdf48d2ff7b4afa8e71df9fba43c66d7 ./tests/amm/test/unit/yieldSpace/pure/hTokenOutForUnderlyingIn.ts

8abdafd6d223847b761277acb18236f2f06bae227fee4eebc8f26531b837479f ./tests/amm/test/unit/yieldSpace/pure/underlyingInForHTokenOut.ts

f9a239f15169f140c8adb15383cfc89c894a140725d2b6f15ce2d5f46ff9ddcf ./tests/amm/test/unit/yieldSpace/pure/underlyingOutForHTokenIn.ts

b6f7329be0c368705ad26ca91897284a7ec3aeee6aad5ceff3c29e6b8ff529f2 ./tests/amm/test/unit/hifiPoolRegistry/HifiPoolRegistry.behavior.ts

191b594ca012ae450829edddba6280e623ec3bee525a709982a3115c8a506f2e ./tests/amm/test/unit/hifiPoolRegistry/HifiPoolRegistry.ts

f4cd1fc7e384665701948205d5020ada12e12b3350d1290764ebe9ad9143fc06 ./tests/amm/test/unit/hifiPoolRegistry/effects/trackPool.ts

777091182af3ab576b18de8d72d12c54fcd6b00a517f56a22667921c02f6a6d2 ./tests/amm/test/unit/hifiPoolRegistry/effects/untrackPool.ts

5ba3906de46cf90d0d4d482e68876816605ef76d4ae672f08bcf5aa1c74aeb7d ./tests/amm/test/unit/hifiPool/HifiPool.behavior.ts

21a69eaac7e1cd8f017a2a305b136f7fb9686faf53630eb9637a3403b7c98964 ./tests/amm/test/unit/hifiPool/HifiPool.ts

c0a4d301973e95fbd2a46d4b5b5755a87a662fe35d5813439dae32eaad2eacdf ./tests/amm/test/unit/hifiPool/view/getNormalizedUnderlyingReserves.ts

be5e711eebf178aac73dbe1da7bf5af984b46fcc9b9e5199e82c4ad3cb5c3931 ./tests/amm/test/unit/hifiPool/view/getVirtualHTokenReserves.ts

b760ac1c87b582dcba4442bf957fd6483f1d8027c654c4b6861cb479e69f21ac ./tests/amm/test/unit/hifiPool/pure/toInt256.ts

8e3b8b5bb491f5bebea516477dba4f4073db2c84933d2052128b3a8595133e05 ./tests/amm/test/unit/hifiPool/effects/burn.ts

c0aafb35511e0a3cda68d2a7bd66393c86db7cee5c80826eb58f8078f793819c ./tests/amm/test/unit/hifiPool/effects/mint.ts

3a1e186e6eb817dd36943e3764fc7bff0cbe46212b22ef816b2c0090aa65f3de ./tests/amm/test/shared/constants.ts

fc8353192f832c8f3239fcff7995fc939aaa02792fc30572dabc53e8e28b3a52 ./tests/amm/test/shared/contexts.ts

52d8548ca67fe2779b40e003780b68daaa41da02370b4cf09f0552f830dd8b64 ./tests/amm/test/shared/deployers.ts

22b556985d05f21341f4ce5dd1123a6c1f2fe22fea277658cabb83590fc178e3 ./tests/amm/test/shared/fixtures.ts

34fb1d88c3d794006566a4578709f2158774f6b65f02f1fbd46e00adc8100c42 ./tests/amm/test/shared/helpers.ts

03f10b723d52991d83d92058b7c43f19e78d3694f58d2f9772aaa1574d17deac ./tests/amm/test/shared/mirrors.ts

5793b0f8f774d4c575eda9c43cd302cd3f2efba0e81fc4ebb779e5c6a5f29516 ./tests/amm/test/shared/mocks.ts

75c315431d135e920909eeeb335043d33509f3d938aa6d2fa9e3821477672cee ./tests/amm/test/shared/types.ts

a9a88f1c6d92f7bcfb99f4d497ad1d7ea368fc53da4176a0b409185d8cb90b59 ./tests/amm/test/integration/index.ts

509b968eb3630e66ad020cdb0f228b71a4456fb5ef9665809574e8f6a708ec1a ./tests/amm/test/integration/hifiPool/HifiPool.behavior.ts

50d7369d4051cd232e2a027f9e772d7bd0e4ac0ab1a7baca254bfb720dfeba1b ./tests/amm/test/integration/hifiPool/HifiPool.ts

db02049eab8971073380501110f16a1a59e0ce8ee24a5e155769948fe1e72384 ./tests/amm/test/integration/hifiPool/effects/buyHToken.ts

0ddb9c569294b4f4a397b56e3e64db9deac88ebb54d0850c3ff829fe22ec2377 ./tests/amm/test/integration/hifiPool/effects/buyUnderlying.ts

2272df7e896600320ce60df19e3cb5c1aa8842702415f44cff1063521b12520f ./tests/amm/test/integration/hifiPool/effects/sellHToken.ts

31a2991ae7b2b4769ed7c8a7042c0ed75b9935aefa69587dde838aaf80739cee ./tests/amm/test/integration/hifiPool/effects/sellUnderlying.ts

Changelog

- 2022-10-21 - Initial report
- 2022-12-18 - Fix Review

About Quantstamp

Quantstamp is a global leader in blockchain security backed by Pantera, Softbank, and Commonwealth among other preeminent investors. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its white glove security and risk assessment services.

The team consists of web3 thought leaders hailing from top organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Many of the auditors hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 250 audits and secured over $200 billion in digital asset risk from hackers. In addition to providing an array of security services, Quantstamp facilitates the adoption of blockchain technology through strategic investments within the ecosystem and acting as a trusted advisor to help projects scale.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Aave, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.