Quantstamp

XY Finance - YBridge

Executive Summary

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

Type	Cross-chain swaps
Timeline	2024-01-29 through 2024-02-05
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	public documentation additional diagrams
Source Code	• https://github.com/XY-Finance/ybridge-contracts-audit #d338539
Auditors	• Valerian Callens Senior Auditing Engineer • Julio Aguilar Auditing Engineer • Shih-Hung Wang Auditing Engineer

Documentation quality	Medium
Test quality	Medium
Total Findings	28 Fixed: 15 Acknowledged: 8 Mitigated: 5
High severity findings	3 Fixed: 3
Medium severity findings	4 Fixed: 1 Acknowledged: 1 Mitigated: 2
Low severity findings	14 Fixed: 8 Acknowledged: 4 Mitigated: 2
Undetermined severity findings	2 Acknowledged: 1 Mitigated: 1
Informational findings	5 Fixed: 3 Acknowledged: 2

Summary of Findings

This audit focused on yBridge, an in-house bridge used by the XY Finance ecosystem to perform cross-chain swaps between supported periphery chains. A network of validators is in charge of securing communication of cross-chain requests via event monitoring and signatures, but also synchronizing on a dedicated settlement chain the accounting of funds in the system. Liquidity providers can earn fees by depositing liquidity on periphery chains to support the cross-chain swaps. Liquidity can only be provided for a limited list of tokens. Depending on the source and final token requested by users for cross-chain swaps, swaps may be required on both chains (source and destination) and executed via a call to whitelisted DEX aggregators. This audit covers the Version 3 of this service where V3 contracts are implementation contracts designed to replace the V2 contracts already deployed on-chain.

As a disclaimer, the "Express fee" feature was presented as being a Work in Progress. For that reason, it should not be considered directly in the scope of this audit, even if it is present in the files covered by this audit.

Regarding the issues, three High-severity issues can cause a loss of funds. Medium-severity issues are mainly related to interaction with price oracles, Denial of Service attack vectors, and the importance of Role Management for a system heavily relying on multiple roles defined on multiple contracts deployed on multiple chains.

Regarding the project quality, the system is well documented and additional documentation was prepared for this audit (diagrams). However, the distinction between deprecated features from previous versions and new features from V3 (such as the express fee) is not clear. Regarding the test suite, even if happy and unhappy paths are covered by the test suite, a limitation in the testing framework (Brownie) does not provide a full overview of the test coverage of the codebase.

Our team frequently interacted with the XY Finance team to clarify the code and its expected behavior. Their active engagement in answering our questions was crucial and greatly assisted in completing the audit.

Fix review update

The client has either fixed, mitigated, or acknowledged the issues in this report. One test was added to the test suite. Additionally, enhancements have been made to the documentation to clarify the behavior of the system.

ID	DESCRIPTION	SEVERITY	STATUS
XYFI-1	Double-Counting of Received Tokens in Cross-Chain Swaps	High	Fixed
XYFI-2	Compromising One Address with the Role `Manager` Is Enough to Drain Funds From the System and Wallets with Active Approvals	High	Fixed
XYFI-3	Refunded Swaps Can Still Be Closed on the Destination Chain	High	Fixed
XYFI-4	Exposure of the System to DoS Attacks	Medium	Fixed
XYFI-5	Absence of Constraint for the Main Roles of the System	Medium	Mitigated
XYFI-6	Risks Related to Using Price Feed Oracles	Medium	Mitigated
XYFI-7	Privileged Roles and Centralization Risks	Medium	Acknowledged
XYFI-8	Missing Input Validation	Low	Mitigated
XYFI-9	Updates of Important Variables Are Not Always Logged	Low	Fixed
XYFI-10	Deploying the Contract `Supervisor` with Duplicate Validators Can Make This Contract Useless	Low	Fixed
XYFI-11	Upgradability	Low	Acknowledged
XYFI-12	Upgradeable Contracts Should Disable the Initialization of the Implementation Contract	Low	Fixed
XYFI-13	Inherited Contracts Not Initialized by the Child Contracts	Low	Fixed
XYFI-14	Errors Can Happen in the Contract `FeeUtility`	Low	Mitigated
XYFI-15	The Single-Chain Swap Flow for Non-Native Tokens Sends Any Attached Native Token to the Aggregator	Low	Fixed
XYFI-16	User Funds May Be Locked if the Destination Chain of a Cross-Chain Swap Is Not Supported or Inactive	Low	Acknowledged
XYFI-17	Use of Solidity `transfer()` function	Low	Acknowledged
XYFI-18	Risk of Gas Griefing for the Actors `ROLE_YPOOL_WORKER` or `ROLE_LIQUIDITY_WORKER`	Low	Acknowledged
XYFI-19	Greedy Contract	Low	Fixed
XYFI-20	DoS Risks of Using `safeApprove()`	Low	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
XYFI-21	Possible Signature Replay in Case of a Chain Fork	• Low	Fixed
XYFI-22	Using the Same Identifier for Distinct Operations May Lead to Cross-Operation Signature Replay in Future Updates	• Informational	Fixed
XYFI-23	Risks Related to Off-Chain Actions Triggered by on-Chain Events	• Informational	Acknowledged
XYFI-24	Cross-Chain Swaps Can Be Triggered when the Source Chain Is Also the Target Chain	• Informational	Fixed
XYFI-25	Initialization Function Does Not Work on Already Deployed Contracts	• Informational	Acknowledged
XYFI-26	Consolidated List of Best Practices	• Informational	Fixed
XYFI-27	Incorrect Updates Possible via the Function `setYBridgeVault()`	• Undetermined	Mitigated
XYFI-28	Partial Visibility over the Current Test Coverage of the System	• Undetermined	Acknowledged

Assessment Breakdown

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

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

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

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. 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:

   1. 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.
   2. 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, clarity, 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.

Scope

The scope was limited to the files in the folder `contracts/*` at the commit `d338539b`, except:

- contracts/periphery-chain/UniERC20.sol
- contracts/periphery-chain/XYWrappedToken.sol
- contracts/settlement-chain/RebalanceRewardCalculation.sol
- contracts/UUPSProxy.sol
- contracts/mock/*

These contracts were in scope:

- Supervisor.sol
- YBridgeV3.sol
- YBridgeVaultV3.sol
- FeeUtility.sol
- YBridgeSettlementV3.sol
- NativeTokenPriceFeedConsumer.sol
- OKXDexAggregatorAdaptor.sol

The "Express fee" feature was presented as being a Work in Progress. For that reason, it should not be considered directly in scope of this audit, even if it is present in the files covered by this audit.

Findings

XYFI-1
Double-Counting of Received Tokens in Cross-Chain Swaps

● High ⓘ Fixed

> **Update**
> The client fixed the issue by setting the swap receiver to the `YBridge`, calculating the `yBalance`, and then sending the received funds to the vault. In this way, the attack described in the report will not work anymore as the code calculates the difference of balance for the bridge instead of the vault.

> **Update**
> Marked as "Fixed" by the client. Addressed in: `3ec953956ff57db8dd3fe3031710cce5e092698b`.

File(s) affected: `YBridgeV3.sol, YBridgeVaultV3.sol`

Description: When a user initiates a cross-chain swap, the source tokens will be swapped to a vault's underlying tokens through a whitelisted DEX aggregator, if necessary. In the `YBridgeV3._swap()` function, the swap logic calculates the balance difference of the underlying token in the vault before and after the swap and considers it as the token amount to be bridged.

However, suppose a user can call `YBridgeVaultV3.deposit()` during the swap, therefore increasing the vault's balance. In that case, the provided underlying tokens will be counted as the deposited and bridged tokens at the same time. As a result, the user will receive the XY-wrapped tokens (representing shares of the vault) at no additional cost.

The above scenario is possible since the user controls the call data passed to the whitelisted aggregators, and the protocol supports the 1inch V5 router. The 1inch V5 router exposes a `swap()` function, which allows the caller to specify the swap `executor` and receive a call from the router during the swap.

Recommendation: Consider setting the receiver of the swap to the bridge contract instead of the vault and transferring the swapped tokens to the vault after calculating the balance difference. This way, the issue is mitigated as depositing into the vault no longer affects the swap results.

XYFI-2
Compromising One Address with the Role `Manager` Is Enough to Drain Funds From the System and Wallets with Active Approvals

• High ⓘ Fixed

Update

Checks were added to restrict the addresses that can be whitelisted as aggregators. As a side note, the first check relies on a correct configuration on the vault. But, technically, the owner of the vault can first call `setYBridge()` to set `yBridge` to an arbitrary address. Then, the manager calls `setAggregator()` to set the vault as an aggregator. Lastly, the owner of the vault calls `setYBridge()` to reset `yBridge`. This attack scenario, however, requires both the manager and the owner of the vault to be compromised, so its likelihood is relatively low.

Update

Marked as "Fixed" by the client. Addressed in: `20a63175eca17d2e0400e057cc24d6f10abb4b6f`.

File(s) affected: `YBridgeV3.sol`

Description: The role `Manager` is used in the `YBridgeV3` to execute restricted operations and can only be assigned by an address owning the role `Owner`. For that reason, it may be subject to fewer protection measures than an address with the role `Owner`. Compromising one address with that role is enough to drain funds from the system and wallets with active approvals for four main reasons:

1. Addresses with the role `Manager` are authorized to whitelist aggregators for the contract `YBridgeV3` via the function `setAggregator()`. This operation does not require any third-party validation.
2. Any user can trigger a single chain swap from the contract `YBridgeV3` via the functions `singleChainSwap()` or `singleChainSwapWithReferrer()`.
3. The function `_singleChainSwap()` performs arbitrary calls to any address that is a whitelisted aggregator, with an arbitrary calldata `aggregatorData` passed by the user triggering the single chain swap.
4. The function `transferTo()` of the contract `YBridgeVaultV3()` can be used to transfer any arbitrary amount or any arbitrary token from the contract to any arbitrary address. It is protected by the modifier `onlyBridge`.

Based on these four items, a compromised Manager can whitelist as aggregators all `YBridgeVaultV3` contracts associated with the bridge and then initiate a single chain swap with the `YBridgeVaultV3` contracts as `aggregator` and a `calldata` value triggering a call to the function `transferTo()` with specific parameters to transfer any arbitrary amount or any arbitrary token from the contract to any arbitrary address, all without any valid signature from validators.

Similar steps can also be used to drain any funds from any wallet with an active token approval for the bridge contract. For this, the `aggregator` to whitelist should be the token to drain, and the `calldata` should represent a call to `transferFrom()`.

Recommendation: Consider preventing from whitelisting a vault as an aggregator or a supported token. Another method could be to check the first bytes of the `aggregatorData` value used in `_singleChainSwap()`. In parallel, consider using security best practices to prevent any compromise of addresses owning the role `Manager`.

XYFI-3
Refunded Swaps Can Still Be Closed on the Destination Chain

• High ⓘ Fixed

Update

The client added a new `revokeCloseSwapForRefund()` function, which will set `everClosed[universalSwapId]` to `true` after validating the signatures. Also, a `CloseSwapRevoked()` event will be emitted at the end of the function, and the validators should only provide the signatures for refunding on the source chain after the emission of the event.

As side notes, the off-chain logic in the validators is out of the scope of this audit, so we are not able to verify if the validator will only publish the refund signatures after `CloseSwapRevoked()`. Also, the signature for `refund()` and `revokeCloseSwapForRefund()` uses the same identifier, `LOCK_CLOSE_SWAP_AND_REFUND_IDENTIFIER`. It is recommended to use unique identifiers for each type of signature as best practice.

Update

Marked as "Fixed" by the client. Addressed in: `68695c16410a9da9ac4d2b15b6563289f27a3494`. The client provided the following explanation:

> Add revokeCloseSwapForRefund() and CloseSwapRevoked event in YBridgeV3. Validators will sign the signatures for Refund on settlement chain only after the CloseSwapRevoked is emitted. This ensures the swap request cannot be executed on the destination chain.

File(s) affected: `YBridgeV3.sol`

Description: The protocol may choose to refund the tokens to users on the source chain if the cross-chain swap fails to fulfill on the destination chain, e.g., the `closeSwap()` call constantly fails. However, currently, there is no way to nullify the signatures for the `closeSwap()` call or mark a swap as invalid. As a result, an executor can still retry `closeSwap()` with the correct signatures even after the user has received a refund.

This may become an issue if an executor role is compromised or considered an untrusted party, which can happen if the executor role is opened for the public in the future.

Recommendation: Consider adding a function allowing a privileged role to set the `everClosed` status of a swap on the destination chain to `true`, before refunding the user on the source chain. The protocol should only refund users after ensuring the swap cannot be fulfilled on the destination chain.

XYFI-4 Exposure of the System to DoS Attacks

● Medium ⓘ Fixed

Update

The client introduced a minimum amount threshold for the first item, and clarified for the second item that the executor will not submit the transaction on the destination chain if the simulation fails.

Update

Marked as "Fixed" by the client. Addressed in: `e995a7cd8a2c7501a6372bff2a1e9ec34974c016`. The client provided the following explanation:

> 2. The try-catch in closeSwap function should prevent the transaction from reverting if there is a malicious swap route on the destination chain.

File(s) affected: `YBridgeV3.sol`

Description: There are at least two ways for malicious actors to disrupt the network and increase the computing activity of network actors (executors and validators), especially on chains where the gas and fees are the cheapest:

1. Trigger cross-chain flows (cross-chain swaps, deposits, withdrawals) with the minimum allowed amount;
2. Trigger cross-chain swaps with malicious ERC tokens requiring a swap on the target chain and specifically reverting when the sender of a transfer of that malicious token is the contract `YBridgeV3`.

Recommendation: Consider:

1. Enforcing minimum amounts of tokens necessary to trigger cross-chain swaps and sufficient fees to make it expensive for an external actor to negatively impact the system, especially on the cheapest chains supported by XY Finance.
2. Validators could also have an off-chain blacklist mechanism to ignore the cross-chain requests triggered by disruptors or when the token on the destination chain leads to an error. Alternatively, a whitelist of tokens could also be added to the source chain, however, this solution may reduce the capabilities of the protocol and increase the gas costs of executing such an operation.

XYFI-5 Absence of Constraint for the Main Roles of the System

● Medium ⓘ Mitigated

Update

The client considered the recommended items and added a check to cover the particular case.

Update

Marked as "Mitigated" by the client. Addressed in: `0f366ec67419efc7e28ecec7d70352cb4c5bd3d9`. The client provided the following explanation:

> 1. Each role's admin is responsible for effective management. In the event of human errors during operations, we will utilize monitor events and ensure the relationships between all roles and addresses through off-chain means.
> 2. We intentionally assign multiple addresses the same role for certain roles in order to increase the throughput of transactions processed by that chain. Overall, we believe that the admin of each role should manage effectively, such as ensuring that old addresses are revoked before granting access to new addresses when replacing addresses.

> 3. Currently, we believe that its admin can revoke the role of the address is sufficient.
> 4. If multiple addresses share the same role, other addresses can also act freely; if an address fails to complete a certain proportion of verifications within a cycle, we will correspondingly reduce the verification rewards for this cycle.

File(s) affected: `YBridgeV3.sol, YBridgeVaultV3.sol, FeeUtility.sol, YBridgeVaultSettlementV3.sol, Supervisor.sol`

Description: Multiple roles are defined in the system on multiple contracts deployed on multiple chains. It includes the validators added to the Supervisor contract.

Any of these roles can be granted, revoked, transferred, or renounced based on specific rules. It means that each of these roles can be owned at a given time by `0` to `n` addresses.

This results in some possible situations when it becomes impossible to run specific operations because the addresses owning this privilege become temporarily unavailable (voluntarily or not) or do not have the role anymore.

In particular, the `Supervisor.setValidator()` function allows privileged roles to add or remove a validator from the validator set. To further prevent manual errors when configuring the validator set, it is recommended to ensure the `threshold` and `validatorsNum` are non-zero, as a `threshold` of zero would allow anyone to pass the `checkSignatures()` check without providing any valid signatures.

Recommendation: Consider assessing if constraints should be enforced in the code for these roles. For instance:

1. Should it be allowed to remain without address owning a given role?
2. Should it be allowed to remain with more than one address owning a given role?
3. Should an update of the role be done in two steps to avoid giving the role to an address owned by no one?
4. Should there be a negative incentive if a role does not behave correctly in a given amount of time?

For the particular case, consider adding a check to ensure that `validatorsNum` and `threshold` are non-zero.

XYFI-6 Risks Related to Using Price Feed Oracles

• Medium ⓘ Mitigated

> **Update**
> A heartbeat mechanism was added to the default native token price update mechanism.

> **Update**
> Marked as "Mitigated" by the client. Addressed in: `77acfdcb41d6bdcb22ae2dd22bab8010f5c0dfb5`. The client provided the following explanation:
>
> > We will regularly update the default native token price as the upper bound for the native token price from the oracle. This ensures that the price does not become too high, resulting in us receiving fewer native tokens as an express fee.

File(s) affected: `NativeTokenPriceFeedConsumer.sol, YBridgeVaultV3.sol`

Description: The system uses two price feeds to know the current price of native tokens and gas on a given chain: `NativeTokenPriceFeedConsumer` and `GasPriceConsumer`. This leads to several risks:

1. Extreme prices can be provided due to an error, a price manipulation attack, or extreme market conditions. These prices could be abnormally high or low;
2. The number of decimals used by the price feed can be updated;
3. The price returned by the price feed may not be strictly positive;
4. A stale price can be provided if the internal parameters to trigger a price feed update are not adapted to its use case (i.e. maximum deviation or heartbeat thresholds' value too high);
5. The price feed can revert;
6. The `NativeTokenPriceFeedConsumer` can forward the price from the price feed or provide a default price which can be updated manually by the contract's manager role. The price update process may also be affected by various chain conditions. For example, if the chain is congested, the price update could be delayed, as the transaction could not be executed in time. The latter opens up the risk of having an old default price which will impact the fees calculated for each operation;
7. As the protocol will be deployed on several L2s, checking the sequencer's uptime when querying price data is crucial. If the sequencer is down, an incorrect or stale price can be used. This vulnerability could cause the protocol to calculate fees with outdated prices. For more details, please refer to Chainlink L2 Sequencer Uptime Feeds document;
8. The `NativeTokenPriceFeedConsumer` contract fetches the price of the native token from Chainlink oracles if supported and configured on the deployed chain. Chainlink oracles have minimum and maximum prices coded in the aggregator contracts. The price update transaction will fail if the new price exceeds the range. For example, the aggregator for the `MATIC/USD` price feed

on Polygon has a hard-coded `minAnswer` of `1000000` (0.01 USD). If MATIC prices fall below 0.01 USD, the oracle will not be updated, causing the protocol to use an incorrect price.

Any of these scenarios would negatively impact the system if they occur.

Recommendation: Consider adding measures to reduce the likelihood of these scenarios, or failure with a dedicated custom error if they occur:

- For items 1) and 8), consider implementing a circuit breaker on the oracle contracts so that the protocol can be paused when the price of the native token changes significantly in a short period. Also, consider actively monitoring the price of the native assets off-chain and react promptly.
- For item 2) specifically, consider calling `decimals()` on the Chainlink oracles to fetch the actual decimals of the returned price to avoid a potential mismatch if the oracle contracts are updated in the future.
- For item 6) specifically, consider adding a similar heartbeat check when calling `_getDefaultNativeTokenPrice()` to ensure an updated value is known and revert the function call if the price becomes stale.
- For item 7) specifically, consider checking the sequencer's uptime feed to avoid using stale prices. An example code can be found in the Chainlink official documentation.

XYFI-7 Privileged Roles and Centralization Risks

● Medium Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> This page in gitbook link explains roles and their associated privileges, as well as the corresponding security management mechanism for address keys.

File(s) affected: `Supervisor.sol, NativeTokenPriceFeedConsumer.sol, OKXDexAggregatorAdaptor.sol, YBridgeV3.sol, YBridgeVaultV3.sol, FeeUtility.sol, YBridgeVaultSettlementV3.sol`

Description: These are the privileged operations identified during the audit:

In `Supervisor.sol`: If enough valid validators (>= threshold) provide a valid signature for the operation:

- A new address can be added to the list of validators;
- A validator can be removed from the list of validators;
- The threshold value can be updated;

In `NativeTokenPriceFeedConsumer.sol`:

- Addresses with the `DEFAULT_ADMIN_ROLE` can:
  - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the role `ROLE_OWNER` can:
  - Grant or revoke this role and the role `ROLE_MANAGER` to any arbitrary address;
  - Renounce this role;
- Addresses with the role `ROLE_MANAGER` can:
  - Update the default native token price to any arbitrary value;
  - Update the Chainlink Native Token price feed to any arbitrary address with any number of decimals;
  - Update the heartbeat of the price feed to any arbitrary value;
  - Renounce this role;

In `OKXDexAggregatorAdaptor.sol`:

- Addresses with the `DEFAULT_ADMIN_ROLE` can:
  - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the role `ROLE_OWNER` can:
  - Grant or revoke this role and the roles `ROLE_STAFF`, `ROLE_SWAPPER` to any arbitrary address;
  - Renounce this role;
- Addresses with the role `ROLE_STAFF` can:
  - Grant the role `ROLE_SWAPPER` to any arbitrary address;
  - Update the value `aggregator` to any arbitrary address;
  - Update the value `approveToAddress` to any arbitrary address;
- Addresses with the role `ROLE_SWAPPER` can:
  - Execute a swap;

In `YBridgeV3.sol`:

- Addresses with the `DEFAULT_ADMIN_ROLE` can:
  - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the role `ROLE_OWNER` can:
  - Grant or revoke this role and the roles `ROLE_MANAGER`, `ROLE_STAFF`, `ROLE_YPOOL_WORKER` to any arbitrary address;
  - Set `YBridgeVault` and the token to any arbitrary address which is a contract;

  - Rescue funds accidentally sent to this contract, except native tokens;
  - Authorize an upgrade of the contract;
  - Renounce this role;
- Addresses with the role `ROLE_MANAGER` can:
  - Set the native token price feed consumer to any arbitrary address;
  - Set the maximum swap amount of a YBridgeVault token to any arbitrary amount;
  - Set the express fee (in USD) to any non-zero arbitrary amount;
  - Set or unset as a DEX aggregator adaptor any arbitrary address which is a contract;
  - Set or unset as a DEX aggregator any arbitrary address that is a contract;
  - Pause or unpause major functions:
    - `closeSwap()`;
    - `refund()`;
    - `collectExpressFee()`;
    - `swap()`;
    - `swapWithReferrer()`;
    - `singleChainSwap()`;
    - `singleChainSwapWithReferrer()`;
  - Set to accept swap requests or not;
  - Set or unset the chains where the express fee is available;
  - Renounce this role;
- Addresses with the role `ROLE_YPOOL_WORKER` can:
  - Fulfill a swap request for a user. It should use a set of valid signatures from valid validators from the Supervisor;
  - Refund user if a swap request cannot be bridged. It should use a set of valid signatures from valid validators from the Supervisor;
  - Collect express fees that accumulate on this chain. It should use a set of valid signatures from valid validators from the Supervisor;
  - Renounce this role;

In `YBridgeVaultV3.sol`:

- Addresses with the `DEFAULT_ADMIN_ROLE` can:
  - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the role `ROLE_OWNER` can:
  - Grant or revoke this role and the role `ROLE_MANAGER`, `ROLE_STAFF`, `ROLE_LIQUIDITY_WORKER` to any arbitrary address;
  - Set the `yBridge` address to any arbitrary address which is a contract;
  - Rescue funds accidentally sent to this contract, except native token, deposit token, or xyWrappedToken;
  - Emergency remove assets to validators' designated hideout; address. It should use a set of valid signatures from valid validators from the Supervisor;
  - Authorize an upgrade of the contract;
  - Renounce this role;
- Addresses with the role `ROLE_MANAGER` can:
  - Set the new gas price consumer address to any arbitrary address which is a contract;
  - Set the maximum and minimum value of yield rate bound to any arbitrary value with at least `10 ** YIELD_RATE_DECIMALS` decimals;
  - Pause or unpause the following functions:
    - `transferTo()`;
    - `completeDeposit()`;
    - `completeWithdraw()`;
    - `deposit()`;
    - `withdraw()`;
  - Activate or disable accept deposit requests;
  - Activate or disable withdrawal requests;
  - Renounce this role;
- Addresses with the role `ROLE_STAFF` can:
  - Set deposit or withdraw gas limit to any arbitrary value;
  - Set the supervisor address (added in the v3 upgrade). The new address must match with the supervisor of the yBridge address;
  - Set the `chainId` variable (added in the v3 upgrade), which should match the `chainId` of the current chain;
  - Renounce this role;
- Addresses with the role `ROLE_LIQUIDITY_WORKER` can:
  - Complete a deposit or withdrawal request by minting XY-wrapped tokens. It should use a set of valid signatures from valid validators from the Supervisor;
  - Collect fees. It should use a set of valid signatures from valid validators from the Supervisor;
  - Renounce this role;
- Address `ybridge` can:
  - Send any arbitrary amount of any arbitrary token (except native tokens) to any arbitrary address;

In `FeeUtility.sol`:

- Addresses with the role `DEFAULT_ADMIN_ROLE` can:

    - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the role `ROLE_OWNER` can:
    - Grant this role and the role `ROLE_SETTLEMENT_WORKER` to any arbitrary address;
    - Revoke this role and the role `ROLE_SETTLEMENT_WORKER` from any arbitrary address;
    - Renounce this role;
    - Authorize an upgrade of the contract;
- Addresses with the role `ROLE_SETTLEMENT_WORKER` can:
    - Renounce this role;
- If enough valid validators (>= threshold) of the Supervisor contract associated with this contract provide a valid signature for the given operation:
    - Update to any address the recipient of the withholding, closeSwapGas, xyDAOReserve, and express fees;
    - Update the xyDAOReserve fee rate to any arbitrary value;
    - Update the mapping associating the L1 chain ID associated with a given L2 chain ID to any couple of IDs;
    - Update if, for a given chain ID, the native token price refers or not to the L1 chain;
    - Update if, for a given chain ID, the closeSwap has a destination chain or not with any value;
    - Update, for a given chain ID, the latest `PriceRoundData` with any value;
    - Update, for a given couple of chain IDs, the protocol fee config with any value;

In `YBridgeVaultSettlementV3.sol`:

- Addresses with the `DEFAULT_ADMIN_ROLE` can:
    - This role is not assigned to any address in the current implementation of the contract;
- Addresses with the `ROLE_OWNER` can:
    - Grant this role and the role `ROLE_SETTLEMENT_WORKER` to any arbitrary address;
    - Revoke this role and the role `ROLE_SETTLEMENT_WORKER` from any arbitrary address;
    - Update the chain PCV total shares;
    - Renounce this role;
    - Authorize an upgrade of the contract;
- Addresses with the `ROLE_SETTLEMENT_WORKER` can:
    - Register a refund request. It should use a set of valid signatures from valid validators from the Supervisor;
- If enough valid validators (>= threshold) of the Supervisor contract associated with this contract provide a valid signature for the given operation:
    - Register a deposit;
    - Register a withdrawal;
    - Register a cross-chain swap initialized;
    - Register a cross-chain swap settled;
    - Register a request to claim fees;
    - Register a request to claim express fees;

Recommendation: These roles and the associated privileges should be documented to users. Also, key management should follow the latest best practices in security.

XYFI-8 Missing Input Validation

• Low ⓘ Mitigated

Update

Only a part of the recommended items were implemented

Update

Marked as "Fixed" by the client. Addressed in: `4023819e5d01bcda465140d6095db1b053993f5d`.

File(s) affected: `NativeTokenPriceFeedConsumer.sol, OKXDexAggregatorAdaptor.sol, Supervisor.sol, FeeUtility.sol, YBridgeVaultV3.sol, YBridgeV3.sol`

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error:

In `NativeTokenPriceFeedConsumer`:

1. In `setChainLinkNativeTokenPriceFeed()`, verify that the new address is valid and that the number of decimals matches the decimals of the price feed.
2. In `setDefaultNativeTokenPrice()`, there is no minimum or maximum amount limit.
3. There is no minimum or maximum value check in the function `setHeartBeat()`.
4. The `constructor` should make sure the address of the owner and manager are not `address(0)`. Also make sure to double-check the deployment script. Otherwise, the wrong address could lead to a re-deployment.

In `OKXDexAggregatorAdaptor`:

1. The `constructor` should make sure the address of the owner and staff are not `address(0)`. Also make sure to double-check the deployment script. Otherwise, the wrong address could lead to a re-deployment.

In `Supervisor`:

1. In the `constructor`, the threshold can be `0`.

In `FeeUtility`:

1. In `updateXyDaoReserveFeeRate()`, there is no minimum or maximum amount limit.
2. In `updateProtocolFeeConfig()`, there is no check that `minProtocolFeeInUSD <= protocolFeeRate <= maxProtocolFeeInUSD` and `srcChainId != dstChainId`.
3. The `initialize()` function of the `FeeUtility` contract should validate the owner and supervisor addresses. Otherwise, a re-deployment might be necessary.

In `YBridgeVaultV3`:

1. In `setYieldRateBound()`, there is no check that `_minYieldRateBound <= _maxYieldRateBound`.
2. In `setCompleteDepositGasLimit()`, there is no minimum or maximum amount limit.
3. In `setCompleteWithdrawGasLimit()`, there is no minimum or maximum amount limit.
4. In `constructor`, no check makes sure that `_depositTokenDecimal` matches `depositToken.decimals()`.
5. The `constructor` does not validate the address of the privileged roles like `owner`.

In `YBridgeV3`:

1. In `setNativeTokenPriceFeedConsumer()`, there is no check that `_consumer` is a contract.
2. In `setMaxVaultTokenSwapAmount()`, there is no minimum or maximum amount limit.
3. In `setExpressFeeInUSD()`, there is no maximum amount limit.

Recommendation: We recommend adding the relevant checks.

XYFI-9
Updates of Important Variables Are Not Always Logged

• Low ⓘ Fixed

Update
Several events were added when core variables are updated.

Update
Marked as "Fixed" by the client. Addressed in: `d93aa5ed84fb97c1e020cc083ca2e58d400ad9da`, `0cfa757568494cd4ee8fedb38816ac9fafdcce2a`.

File(s) affected: `NativeTokenPriceFeedConsumer.sol, OKXDexAggregatorAdaptor.sol, Supervisor.sol, FeeUtility.sol, YBridgeVaultV3.sol, YBridgeV3.sol, YBridgeVaultSettlementV3.sol`

Description: At multiple locations in the codebase, important state variables can be updated via dedicated functions. For instance, addresses that should receive funds from the contract, price feeds, or updates of validators and thresholds in the contract `Supervisor`. However, an event is not always emitted. Protocol activities should be logged extensively (especially, actions leading to funds transfers or updates of critical parameters) so that external observers can get in real-time a complete and accurate representation of the current protocol state.

Recommendation: Consider assessing for each state variable if an update should be logged and if a range check must be done. Update the code accordingly.

XYFI-10
Deploying the Contract `Supervisor` with Duplicate Validators Can Make This Contract Useless

• Low ⓘ Fixed

Update
The client added a check making sure that duplicated items cannot be added anymore.

Update
Marked as "Fixed" by the client. Addressed in: `6736b0c56603ba26b9ecfc00ae352619f336e23a`.

File(s) affected: `Supervisor.sol`

Description: In the `constructor` of the contract, if the list `_validators` contains duplicated values, it would corrupt the `threshold` value since it would be higher than the exact number of distinct validators. Hence, it is impossible to reach the threshold or update its value.

Recommendation: Make sure there are no duplicates by checking that the items in the list `_validators` are ordered and distinct.

XYFI-11 Upgradability

Low ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client. The client provided the following explanation:

> Acknowledged. Only the owner will execute the upgrade permission, and the owner consists of the XY validators' cold wallet.

File(s) affected: `FeeUtility.sol, YBridgeVaultSettlementV3.sol, YBridgeV3.sol, YBridgeVaultV3.sol`

Description: While upgradability is not a vulnerability in itself, users should be aware that all mentioned contracts can be upgraded at any given time. This audit does not guarantee the behavior of future contracts that they may be upgraded to.

Recommendation: The fact that the contracts can be upgraded and reasons for future upgrades should be communicated to users beforehand.

XYFI-12 Upgradeable Contracts Should Disable the Initialization of the Implementation Contract

Low ⓘ Fixed

Update
The client added a call to `_disableInitializers()` in all related contracts.

Update
Marked as "Fixed" by the client. Addressed in: `29e909a59096907e6a5c5573be70df70ed0be6ad`.

File(s) affected: `FeeUtility.sol, YBridgeVaultSettlementV3.sol, YBridgeV3.sol, YBridgeVaultV3.sol`

Description: The implementation contracts behind a proxy can be initialized by any address via the function `initialize()`. Taking ownership of implementation contracts can open the door to other attack vectors.

Recommendation: When deploying the contract, consider making sure that the implementation contract is initialized within the same transaction to prevent any external user from interacting with it. An option is to use `_disableInitializers()` in the constructor as described here: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract.

XYFI-13 Inherited Contracts Not Initialized by the Child Contracts

Low ⓘ Fixed

Update
The client added a call to the `OZ __XXX_init()` functions in the `initialize()` function of the child contract.

Update
Marked as "Fixed" by the client. Addressed in: `87849213426972a0eb28b050eea6323a0705f874`.

File(s) affected: `FeeUtility.sol, YBridgeVaultSettlementV3.sol, YBridgeV3.sol, YBridgeVaultV3.sol`

Description: The implementation contracts behind a proxy should be initialized via the function `initialize()`, which should trigger the initialization of parent contracts.

Recommendation: Consider making sure that all parent contracts are correctly initialized from the function `initialize()` of child contracts.

XYFI-14 Errors Can Happen in the Contract `FeeUtility`

• Low ⓘ Mitigated

Update
The client fixed items 1/3 and 2/3.

Update
Marked as "Fixed" by the client. Addressed in: `63a7fcd49c54e1654eeed4cf71b8aa096568d9aa` .

File(s) affected: `FeeUtility.sol`

Description: Three main errors can happen in the contract `FeeUtility` .

1. An underflow can happen in the function `getLastTwoRoundByTimestamp()` in the specific situation when `srcTxTimestamp > 0` and no round has been published yet. It would lead to an underflow error when the expression `roundData.roundId - 1` is executed.
2. An underflow can happen in the function `binarySearchRoundId()` in the specific situation when `endId < beginId` . It would lead to an underflow error when the expression `len = high - low` is executed.
3. A stack overflow error could theoretically happen if the function `binarySearchRoundId()` recursively calls itself too many times.

Recommendation: Consider adapting this contract to cover the potential edge cases 1) and 2). Fuzzing could also be used to make sure that the functions `getLastTwoRoundByTimestamp()` and `binarySearchRoundId()` always return the expected value. For the potential edge case 3), a way to prevent this would be to perform the search iteratively with a while loop in a single function, instead of recursively via consecutive recursive calls.

XYFI-15 The Single-Chain Swap Flow for Non-Native Tokens Sends Any Attached Native Token to the Aggregator

• Low ⓘ Fixed

Update
The client added the recommended item.

Update
Marked as "Fixed" by the client. Addressed in: `2ed6a7251b57b80a82e2485ee420f31fd97d02ee` .

File(s) affected: `YBridgeV3.sol`

Description: The function `_singleChainSwap()` does not make sure that `msg.value == 0` for single-chain swaps of non-native tokens and, as a result, will send any attached native token to the aggregator. This scenario is unlikely though.

Recommendation: Consider checking that `msg.value == 0` for single-chain swaps of non-native tokens.

XYFI-16 User Funds May Be Locked if the Destination Chain of a Cross-Chain Swap Is Not Supported or Inactive

• Low ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client. The client provided the following explanation:

> We have reminded our partners in developer doc that when integrating with our contracts and APIs, they should ensure that their destination chain is one of the supported chains.
> - reminder for api user: link
> - reminder for contract user: link
>   Furthermore, in the event that a chain needs to be temporarily paused (e.g., for routine updates), we will simultaneously notify our users through our website and our partners through group channels. This will inform them of the latest status regarding supported chains to prevent their funds from being temporarily locked in our system. If the user's funds are locked within the system due to the chain being suspended, the assets can be refunded to the user through the refund function, returning their funds on the source chain.

File(s) affected: `YBridgeV3.sol`

Description: Users can trigger cross-chain swaps from a source chain to a destination chain via the functions `swap()` or `swapWithReferrer()`. However, their funds may remain locked on the source chain if the destination chain of a cross-chain swap is not supported or currently paused, though the likelihood of this scenario is limited.

Recommendation: Consider preventing this from happening on-chain or documenting to users the fact that they should use a supported and active destination chain when triggering a cross-chain swap.

XYFI-17 Use of Solidity `transfer()` function

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> We anticipate that fee receivers will all be EOAs, and to save on gas costs, we will maintain the use of transfer().

File(s) affected: `YBridgeVaultV3.sol`

Description: Solidity's `transfer()` function forwards a fixed amount of 2300 gas to the recipient when transferring native tokens to them, which causes several limitations if the recipient is a contract. First, the limited amount of gas prevents the contract from executing more complicated logic (e.g., external calls) in its `fallback()` or `receive()` function. Second, the gas costs of each opcode are subject to change; therefore, it is not guaranteed that currently successful transfers will not fail in the future.

The `YBridgeVaultV3.collectFees()` function uses `transfer()` to transfer native tokens to `withholdingFeesReceiver`.

Recommendation: Consider using `call{value: value_}("")` to transfer native tokens to the recipient. However, note that it is a vector for reentrancy attacks and it should be used in parallel with the CEI pattern to prevent any harm to your system.

XYFI-18 Risk of Gas Griefing for the Actors `ROLE_YPOOL_WORKER` or `ROLE_LIQUIDITY_WORKER`

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> Worker addresses have been added to the GitBook. Additionally, if the admin detects any malicious behavior from workers, their role will be revoked.

File(s) affected: `YBridgeV3.sol, YBridgeVaultV3.sol`

Description: The function `closeSwap()` can be called by an address with the role `ROLE_YPOOL_WORKER`. It can result in transfers to an external address `receiver`. This address can maliciously spend an abnormally high amount of gas that will have to be paid by the address executing this transaction. It is also the case for actors with the role `ROLE_LIQUIDITY_WORKER` in the function `completeWithdraw()` of `YBridgeVaultV3`.

Recommendation: Consider documenting the fact that addresses with the role `ROLE_YPOOL_WORKER` or `ROLE_LIQUIDITY_WORKER` must make sure that the operations are not a griefing attack before executing it.

XYFI-19 Greedy Contract

• Low ⓘ Fixed

Update

The client removed the function `receive()`.

Update

Marked as "Fixed" by the client. Addressed in: `2b7b2eea489d1d64fad0e19af2fbe44b9336e3a9`.

File(s) affected: `FeeUtility.sol`

Description: A greedy contract is a contract that can receive native tokens which can never be redeemed. It is the case of the contract `FeeUtility`.

Recommendation: Consider adding a function to remove native tokens from the contract or removing the function `receive()`.

XYFI-20 DoS Risks of Using `safeApprove()`

• Low ⓘ Fixed

Update
The client added the recommended fix.

Update
Marked as "Fixed" by the client. Addressed in: `c0a514e8e01ce919a9e81e51bbddb6a450ce7ebc`.

File(s) affected: `OKXDexAggregatorAdaptor.sol`

Description: The `OKXDexAggregatorAdaptor` and `YBridgeV3` uses `safeApprove()`, defined in the OpenZeppelin library, to approve another contract to transfer funds from them during a swap.

However, the use of `safeApprove()` in `OKXDexAggregatorAdaptor` may cause a DoS issue. The `safeApprove()` function checks that the caller is either setting their allowance to zero or setting it from zero. Therefore, the call will fail when the allowance is set from a non-zero value to another non-zero value. Consider the below exploit scenario:

1. During the `OKXDexAggregatorAdaptor.swap()` call, the contract approves the DEX aggregator contract with an `amount` of tokens, say 100 USDC.
2. However, the provided swap `data` only causes 99 USDC to be transferred from the contract to the DEX aggregator. Recall that the swap `data` is unrestricted and can have arbitrary content.
3. As a result, the allowance is reduced to 1 USDC after the call to the aggregator.
4. The next time a transaction calls `safeApprove()`, the call will fail since the existing allowance is non-zero (unless the new `amount` is 0).

Recommendation: Consider calling `safeApprove(address, 0)` after calling the DEX aggregator to reset the allowance, as done in the `YBridgeV3` contract.

XYFI-21 Possible Signature Replay in Case of a Chain Fork

• Low ⓘ Fixed

Update
The client added that capability to all contracts.

Update
Marked as "Fixed" by the client. Addressed in: `dc05683371e5906a434ce29c62e3a267c7d6208a`.

File(s) affected: `Supervisor.sol, YBridgeV3.sol, YBridgeVaultV3.sol`

Description: The `Supervisor` and `YBridgeV3` contracts are not able to update their state variable `chainId`. The `YBridgeVaultV3` has the setter function `setChainId()` which only allows a change if `chainId` was not already initialized. In case of a chain fork, the deployed contracts will live in both the old and new chains with duplicated state variables (including the old value of `chainId` in the vault). Not updating the `chainId` could result in the vaults getting fully drained by re-using signatures since both have the same `chaindId`. However, this would only be possible if the workers were not owned by the protocol, which is not the case at the moment, but could be in the future, as stated by the XY Finance team.

Recommendation: Consider adding the capability to update the `chainId` in these three contracts.

XYFI-22 Using the Same Identifier for Distinct Operations May Lead to Cross-Operation Signature Replay in Future Updates

• Informational ⓘ Fixed

Update
The client added a comment to the functions.

Update
Marked as "Fixed" by the client. Addressed in: `9c1df4682cb3dac310d5fe3e88c679156ba591bf` .

File(s) affected: `FeeUtility.sol`

Description: The system heavily relies on signatures to perform critical operations. One way to prevent signature replays between the distinct functions in the same contract is to use a dedicated operation identifier for each function. However, the same identifier `UPDATE_L1_CHAINS` is used in `updateL1Chains()` and `setIsNativeTokenPriceReferToL1()` . Since the function has the same number of parameters and almost the same order of variable types, an incorrect small update of this code could make it possible to use the signatures to validate a call to one function for the other function.

Recommendation: Consider using a distinct identifier for these two functions, or adding a comment saying that the types used for the message should be carefully checked in case of an update.

XYFI-23
Risks Related to Off-Chain Actions Triggered by on-Chain Events

• Informational ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client. The client provided the following explanation:

> Add a session to the yBridge consensus page in gitbook link to explain that validators will only listen to specific events emitted by specific contracts and deemed valid by their observation script.

Description: The system heavily uses on-chain events emitted by the contracts as triggers for cross-chain transfers of value. If the way to detect events is not done correctly, off-chain observers could be tricked by malicious users and consider invalid events as valid ones. This could happen for instance if the event to be emitted by contract `C` is emitted by another contract than `C` , within a transaction also involving `C` or not. Even if the root cause is not exactly the same, being exposed to false events could have the same consequences as what happened for the bridge Qubit.

Recommendation: Consider documenting the fact that validators should make sure that only specific events emitted by specific contracts should be considered valid by their observation script.

XYFI-24
Cross-Chain Swaps Can Be Triggered when the Source Chain Is Also the Target Chain

• Informational ⓘ Fixed

Update
The client added a check to the function `_swap()` .

Update
Marked as "Fixed" by the client. Addressed in: `1aec9bc5828744542b3702bb92946537f041d53b` .

File(s) affected: `YBridgeV3.sol`

Description: Cross-chain swaps can be triggered when the source chain is also the target chain via the functions `swap()` and `swapWithReferrer()` . Since another flow exists for single-chain swaps and cross-chain swaps are subject to more fees, this scenario should be prevented by an error.

Recommendation: Consider preventing cross-chain swaps when the source chain is also the target chain.

XYFI-25 Initialization Function Does Not Work on Already Deployed Contracts

• Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> Acknowledged. Currently, we have no situations where we need to use the reinitializer.

File(s) affected: `YBridgeV3.sol, YBridgeVaultV3.sol, YBridgeVaultSettlementV3.sol`

Description: The `YBridgeVaultV3` contract initializes the new state variable `chainId` inside the `initialize()` function that has the `initializer()` modifier. However, a previous version of this contract is already deployed behind a proxy which means that the modifier cannot be called again, and the variable will be left uninitialized when upgrading the contract.

Recommendation: Fortunately, there is a setter function for the `chainId` that can be called right after upgrading the contract. However, for future upgrades, consider using a newer version of the `Initializable` contract that allows to re-initialize contracts through the `reinitializer()` modifier.

XYFI-26 Consolidated List of Best Practices

• Informational ⓘ Fixed

Update

The client fixed most of the recommended items. Since these are best practices, we consider this issue Fixed.

Update

Marked as "Fixed" by the client. Addressed in: `581064ebc51f722315c6862f5670eaca93448ce2` , `f9b51b29a819969da5bc78452c55da15ec8df682` .

File(s) affected: `NativeTokenPriceFeedConsumer.sol, OKXDexAggregatorAdaptor.sol, Supervisor.sol, FeeUtility.sol, YBridgeVaultV3.sol, YBridgeV3.sol, YBridgeVaultSettlementV3.sol`

Description: In `NativeTokenPriceFeedConsumer.sol` :

1. Consider adding a visibility keyword to `heartBeat` .
2. Consider using the `constant` keyword for `usdValueDecimals` .
3. The function `_getDefaultNativeTokenPrice()` can be replaced with a direct state variable access.

In `OKXDexAggregatorAdaptor.sol` :

1. `SafeMath` is imported but not used.
2. Only `IERC20` could be imported instead of `ERC20` .

In `Supervisor.sol` :

1. In the `contract` , the state variable `chainId` can be replaced with a utility function `getChainId()` using the low-level instruction `chainId()` ;
2. Consider caching the value of `_validators.length` in a local variable to save gas;

In `FeeUtility.sol` :

1. `Address` is imported but not used.
2. The role `ROLE_SETTLEMENT_WORKER` is declared but never used.
3. Consider caching the `PriceRoundData` object in a local variable in the function `updatePriceRoundData()` to save gas;

In `YBridgeVaultV3.sol` :

1. Only `IERC20` could be imported instead of `ERC20` .
2. In the function `rescue()` , consider caching in a local variable the value of `tokens.length` to save gas.
3. In the function `completeDeposit()` , consider caching in a local variable the value of `vaultTokenAmount * 10 ** (YIELD_RATE_DECIMALS + XY_TOKEN_DECIMALS - depositTokenDecimal) / shareAmount` to save gas.

In `YBridgeV3.sol` :

1. Only `IERC20` could be imported instead of `ERC20` .
2. In the function `_swap()` , consider caching in a local variable the value of `YBridgeVaults[address(swapDesc.dstToken)]` to save gas.
3. In the function `rescue()` , consider caching in a local variable the value of `tokens.length` to save gas.

In `YBridgeVaultSettlementV3.sol` :

1. In the function `settleCrossChainSwap()`, consider caching in a local variable the value of `_swapInfo[universalSwapId]` to save gas.
2. In the function `claimFees()`, consider caching in a local variable the value of `feeUtility.withholdingFeeReceiver()`, `feeUtility.closeSwapGasFeeReceiver()` and `feeUtility.xyDaoReserveFeeReceiver()` to save gas.

Recommendation: Consider following these best practices.

XYFI-27
Incorrect Updates Possible via the Function `setYBridgeVault()`

• Undetermined ⓘ Mitigated

Update

We consider this issue as mitigated. We can still override a given active vault association by executing the following two steps:

1. `setYBridgeVault(tokenA, vaultXForA, true)`
2. `setYBridgeVault(tokenA, vaultYForA, true)` After step 2, we could still have tokens in `vaultXForA`.

As mitigation measures, the client identified the following operations to be performed before doing such update:
1. Cease the acceptance of swap requests for vault tokenA on the destination chain.
2. Clear swap, deposit, and withdraw requests for vault tokenA.
3. Set the status of the old yBridgeVault as False on the yBridge contract.
4. Remove tokenA liquidity from the old yBridgeVault.
5. Update validators to invalidate the old yBridgeVault.
6. Update validators to validate new yBridgeVault requests.
7. Set the new yBridgeVault for tokenA on the yBridge contract.
8. Resume normal operations.

Update

Marked as "Fixed" by the client. Addressed in: `15027c28ee36dbe6985198bb2eff6e4c2e944a11`.

File(s) affected: `YBridgeV3.sol`

Description: The function `setYBridgeVault()` can be called to support a given token and assign its associated `YBridgeVault`. However, no check in this function prevents from:

1. updating the underlying vault associated with a given token;
2. having more than one token pointing to the same vault, even if a vault can only support one token;

In the best-case scenario, this could block users from requesting a cross-chain swap since the check if `(!vaultSupportedToken[address(swapDesc.dstToken)]) {revert InvalidVaultToken();}` would cause the function to revert. However, in the worst-case scenario, the wrong tokens might be sent to the vault, leading to a bad accounting in the settlement layer.

Since these functions can only be called by an address with the role `ROLE_OWNER`, the likelihood for this to happen is limited.

Recommendation: Consider creating separate functions to activate and deactivate the support of a token.
1. A token should not be supported if it is already supported.
2. A token should only be associated with a vault having this token as `depositToken`.
3. A vault should not be associated with a token if it is already associated with another token.
4. Deactivating a token should lead to removing its association with the vault.

XYFI-28
Partial Visibility over the Current Test Coverage of the System

• Undetermined ⓘ Acknowledged

Update

The client identified the cause of the issue

Update

Marked as "Unresolved" by the client. The client provided the following explanation:

> It seems that the issue arises from Brownie's inability to find the corresponding implementation function when fetching the proxy contract.

Description: The framework used to test the system is Brownie. However, running the command `brownie --test coverage` gives an incomplete result where only some contracts are listed, even if tests are defined for all contracts and all tests passed. Only the contracts `NativeTokenPriceFeedConsumer`, `OKXDexAggregatorAdaptor`, and `Supervisor` are listed. The others are not (`YBridgeV3`, `YBridgeVaultV3`, `FeeUtility`, `YBridgeVaultSettlementV3`). As a result, it is not possible to identify which parts of the system are covered or not by the test suite for these specific contracts containing most of the business logic.

Recommendation: Consider fixing the issue causing incomplete coverage results. Alternatively, consider using Foundry instead of Brownie.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

Toolset

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

Setup

Tool Setup:

- Slither

Steps taken to run the tools:

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

Automated Analysis

Slither

We filtered the issues highlighted by Slither and consolidated the relevant issues in this report.

Test Suite Results

Even if happy and unhappy paths are covered by the test suite, the execution resulted in several warnings.

```
tests/unit_test/test_fee_utility.py ............ [ 14%]
tests/unit_test/test_supervisor.py .......... [ 27%]
tests/unit_test/test_ybridge_v3.py ................... [ 50%]
```

```
tests/unit_test/test_ybridge_vault_settlement_v3.py ....................... [ 80%]
tests/unit_test/test_ybridge_vault_v3.py ............... [100%]

=========================================================== warnings summary
============================================================
tests/unit_test/test_fee_utility.py: 199 warnings
tests/unit_test/test_ybridge_v3.py: 35 warnings
tests/unit_test/test_ybridge_vault_settlement_v3.py: 126 warnings
tests/unit_test/test_ybridge_vault_v3.py: 15 warnings
tests/unit_test/test_ybridge_v3.py: 64 warnings
tests/unit_test/test_ybridge_vault_settlement_v3.py: 43 warnings
tests/unit_test/test_ybridge_vault_v3.py: 24 warnings
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth

============================================= 81 passed, 510 warnings in 273.89s (0:04:33)
==============================================

**Fix Review Update**

tests/unit_test/test_fee_utility.py ...........
[ 14%]
tests/unit_test/test_supervisor.py ..........
[ 26%]
tests/unit_test/test_ybridge_v3.py ...................
[ 51%]
tests/unit_test/test_ybridge_vault_settlement_v3.py .......................
[ 80%]
tests/unit_test/test_ybridge_vault_v3.py ...............
[100%]

=========================================== warnings summary
===========================================
tests/unit_test/test_fee_utility.py: 199 warnings
tests/unit_test/test_ybridge_v3.py: 64 warnings
tests/unit_test/test_ybridge_vault_settlement_v3.py: 126 warnings
tests/unit_test/test_ybridge_vault_v3.py: 15 warnings
tests/unit_test/test_ybridge_v3.py: 116 warnings
tests/unit_test/test_ybridge_vault_settlement_v3.py: 43 warnings
tests/unit_test/test_ybridge_vault_v3.py: 24 warnings
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth
tests/unit_test/test_ybridge_v3.py::test_swap_with_vault_token_eth

=================================== 82 passed, 591 warnings in 24.27s
===================================
```

Code Coverage

As mentioned in the associated issue, the command `brownie --test coverage` gives an incomplete result where only some contracts are listed, even if tests are defined for all contracts and all tests passed. Only the contracts `NativeTokenPriceFeedConsumer`, `OKXDexAggregatorAdaptor`, and `Supervisor` are listed. The others are not (`YBridgeV3`, `YBridgeVaultV3`, `FeeUtility`, `YBridgeVaultSettlementV3`).

```
================================================== Coverage
==================================================
  contract: MockContract - 37.3%
    MockContract.updateInvocationCount - 75.0%
    MockContract.fallbackImpl - 50.8%
    MockContract.givenAnyReturnBool - 0.0%
    MockContract.givenCalldataReturnBool - 0.0%
    MockContract.givenMethodReturnBool - 0.0%
    MockContract.trackCalldataMock - 0.0%
    MockContract.trackMethodIdMock - 0.0%
  contract: NativeTokenPriceFeedConsumer - 15.5%
    NativeTokenPriceFeedConsumer.getLatestNativeTokenPrice - 58.3%
    AccessControl._checkRole - 25.0%
    AccessControl._grantRole - 0.0%
    AccessControl._revokeRole - 0.0%
    AccessControl.renounceRole - 0.0%
    NativeTokenPriceFeedConsumer._getLatestNativeTokenPriceFromPriceFeed - 0.0%
  contract: OKXDexAggregatorAdaptor - 7.1%
    AccessControl._grantRole - 75.0%
    AccessControl._checkRole - 25.0%
    AccessControl._revokeRole - 0.0%
    AccessControl.renounceRole - 0.0%
    Address.functionCallWithValue - 0.0%
    Address.verifyCallResult - 0.0%
    OKXDexAggregatorAdaptor.swap - 0.0%
  contract: Supervisor - 62.8%
    Supervisor.setThreshold - 93.8%
    Supervisor.checkSignatures - 91.7%
    Supervisor.setValidator - 79.2%
    ECDSA.tryRecover - 37.5%
    ECDSA._throwError - 5.0%
  contract: TestAggregator - 70.8%
    TestAggregator.buy - 70.8%
  contract: TestERC20 - 68.2%
    ERC20._transfer - 83.3%
    ERC20._approve - 75.0%
    ERC20._mint - 75.0%
    ERC20.transferFrom - 75.0%
    ERC20.decreaseAllowance - 0.0%
  contract: UUPSProxy - 0.0%
  contract: XYWrappedToken - 65.7%
    ERC20._approve - 75.0%
    ERC20._burn - 75.0%
    ERC20._mint - 75.0%
    ERC20._transfer - 75.0%
    ERC20.transferFrom - 75.0%
    Ownable.transferOwnership - 75.0%
    XYWrappedToken.mint - 50.0%
    ERC20.decreaseAllowance - 0.0%

View the report using the Brownie GUI
============================= 81 passed, 1231 warnings in 3970.87s (1:06:10)
============================

**Fix review update**

=============================================== Coverage
===============================================

  contract: MockContract - 32.3%
    MockContract.fallbackImpl - 50.8%
    MockContract.updateInvocationCount - 50.0%
    MockContract.givenAnyReturnBool - 0.0%
    MockContract.givenCalldataReturnBool - 0.0%
    MockContract.givenMethodReturnBool - 0.0%
    MockContract.trackCalldataMock - 0.0%
    MockContract.trackMethodIdMock - 0.0%

  contract: NativeTokenPriceFeedConsumer - 24.9%
    NativeTokenPriceFeedConsumer._getDefaultNativeTokenPrice - 75.0%
    AccessControl._checkRole - 50.0%
    NativeTokenPriceFeedConsumer.getLatestNativeTokenPrice - 46.7%
    AccessControl._grantRole - 0.0%
    AccessControl._revokeRole - 0.0%
    AccessControl.renounceRole - 0.0%
    NativeTokenPriceFeedConsumer._getLatestNativeTokenPriceFromPriceFeed - 0.0%

  contract: OKXDexAggregatorAdaptor - 61.9%
    UniERC20.uniBalanceOf - 100.0%
    AccessControl._grantRole - 75.0%
    Address.functionCallWithValue - 75.0%
    OKXDexAggregatorAdaptor.swap - 75.0%
    SafeERC20._callOptionalReturn - 75.0%
    AccessControl._checkRole - 50.0%
    Address.verifyCallResult - 37.5%
    AccessControl._revokeRole - 0.0%
    AccessControl.renounceRole - 0.0%

  contract: Supervisor - 55.7%
    Supervisor.setThreshold - 93.8%
    Supervisor.checkSignatures - 91.7%
    Supervisor.setValidator - 78.1%
    ECDSA.tryRecover - 37.5%
    ECDSA._throwError - 5.0%
    Supervisor.setChainId - 0.0%

  contract: TestAggregator - 100.0%
    TestAggregator.buy - 100.0%

  contract: TestERC20 - 68.2%
    ERC20._transfer - 83.3%
    ERC20._approve - 75.0%
    ERC20._mint - 75.0%
    ERC20.transferFrom - 75.0%
    ERC20.decreaseAllowance - 0.0%

  contract: UUPSProxy - 0.0%

  contract: XYWrappedToken - 67.9%
    ERC20._approve - 75.0%
    ERC20._burn - 75.0%
    ERC20._mint - 75.0%
    ERC20._spendAllowance - 75.0%
    ERC20._transfer - 75.0%
    Ownable._checkOwner - 75.0%
    Ownable.transferOwnership - 75.0%
    XYWrappedToken.mint - 50.0%
    ERC20.decreaseAllowance - 0.0%

View the report using the Brownie GUI
============================ 82 passed, 1313 warnings in 811.28s (0:13:31)
=============================
```

Changelog

- 2024-02-06 - Initial report
- 2024-03-05 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers 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 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that

Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

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, 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

Notice of confidentiality

Links to other websites

Disclaimer