Quantstamp

Solv Protocol - SolvBTC

Executive Summary

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

Type	Yield Protocol
Timeline	2024-07-09 through 2024-07-17
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Overview
Source Code	• solv-finance/SolvBTC #eafc741
Auditors	• Adrian Koegl Auditing Engineer • Shih-Hung Wang Auditing Engineer • Cameron Biniamow Auditing Engineer

Documentation quality	Low
Test quality	Medium
Total Findings	7 Fixed: 3 Acknowledged: 3 Mitigated: 1
High severity findings	0
Medium severity findings	1 Mitigated: 1
Low severity findings	4 Fixed: 2 Acknowledged: 2
Undetermined severity findings	0
Informational findings	2 Fixed: 1 Acknowledged: 1

Summary of Findings

Quantstamp has completed an audit of the contracts to be upgraded in the Solv Protocol. These contracts enable the Solv Protocol to represent multiple `ERC3525` assets/slots using a single ERC20 token.

The audited contracts include:

- `SolvBTC` : These new ERC20 tokens represent several selected assets and will replace `SftWrappedToken` (not in scope).
- `SolvBTCYieldTokenOracleForSFT.sol` : This contract provides NAV (Net Asset Value) calculation and oracle management for `SolvBTCYieldToken.sol` , an extension of `SolvBTC` , associated with SFTs.
- `SolvBTCMultiAssetPool` : This contract stores the selected assets and issues `SolvBTC` tokens.
- `SftWrapRouter` : This user-facing contract facilitates the staking and unstaking of ERC3525 assets and routing operations to the appropriate contract.
- `SftWrappedTokenFactory` : This contract manages the deployment and upgrading of beacons and proxies for wrapped tokens, ensuring efficient and flexible contract upgrades.

Overall, the code is well-written, and the Solv team has shown excellent engineering skills. However, the audit process was somewhat hindered by the lack of technical documentation, making it challenging to fully understand the upgrade context and intentions.

Fix Review Update: The Solv team has reworked their contracts, fixed 3 issues, and acknowledged 3 issues. Most of our suggestions were implemented. It should be noted that some of the reworked code was not in scope for the fix review since it did not directly address an issue or suggestions. However, we were able to identify an out-of-scope issue that we recommend addressing.

Fix Review Update 2: The new issue SOLV-7 and additions in SOLV-10 were acknowledged by the client.

Fix Review Update 3: The client added a new `burn()` function for compatibility with Chainlink CCIP. Another role, `SOLVBTC_POOL_BURNER` , was added that can burn unrestricted amounts of `SolvBTC` tokens from any address.

ID	DESCRIPTION	SEVERITY	STATUS
SOLV-1	**Attacker Can Drain Value From Contract**	Medium	Mitigated
SOLV-2	**Oracle May Return Outdated Nav Information**	Low	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
SOLV-3	Inability to Stake Non-Legacy SFT Slots	Low	Fixed
SOLV-4	Inability to Restrict SFT Slot Deposits	Low	Fixed
SOLV-5	Unimplemented Interfaces in `SolvBTCMultiAssetPool`	Informational	Acknowledged
SOLV-6	Disconnected Upgrade Process	Informational	Fixed
SOLV-7	Potential Mismatch in `beacon` Contract	Low	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. The SftWrappedToken and OpenFundMarket contracts were out of scope for this audit.

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

We have audited the components of the Solv Protocol that will be upgraded. The scope specifically excludes these files: `SolvWrappedToken.sol` and `OpenFundMarket.sol`

Files Included

Repo: https://github.com/solv-finance/Solv-Yield-Bearing-Tokens/tree/features/multi-token

Files: `contracts/SftWrappedTokenFactory.sol` `contracts/SolvBTC.sol` `contracts/SolvBTCYieldToken.sol` `contracts/SolvBTCMultiAssetPool.sol` `contracts/SftWrapRouter.sol` `contracts/oracle/SolvBTCYieldTokenOracleForSFT`

Files Excluded

Repo: https://github.com/solv-finance/Solv-Yield-Bearing-Tokens/tree/features/multi-token

Files: `SolvWrappedToken.sol` `OpenFundMarket.sol`

Operational Considerations

- We assume all contracts not in scope to be secure.
- The `SolvBTCMultiAssetPool` contract is designed to be the minter of the `SolvBTC` contract. The protocol team should be cautious when granting any other EOA or contract with a minter role, as a compromised minter may allow arbitrary minting or burning of `SolvBTC`.
- The oracle `SolvBTCYieldTokenOracleForSFT` depends on another external oracle, `SFTNavOracle`, to fetch the NAV of an ERC-20 token. The external oracle is out of the scope of the audit. The protocol team should ensure that the external oracle works as expected and is resistant to manipulation. Otherwise, the protocol may use an incorrect price.
- The `SolvBTC`, `SolvBTCMultiAssetPool`, `SolvBTCYieldToken`, `SftWrapRouter`, and `SolvBTCYieldTokenOracleForSFT` contracts are upgradeable and can be upgraded at any time by the contract owner. This audit does not guarantee the behavior of future contract upgrades of these contracts. We assume all future upgrades will go through thorough audits.

Key Actors And Their Capabilities

1. `SftWrappedTokenFactory`
    1. `admin`
        1. Can update and upgrade the beacon `implementation` contract address for any `productType`.
        2. Can transfer the ownership of any `productType` beacon contract.
    2. `governor`
        1. Can deploy a product proxy contract using the preset product type beacon contract as its implementation.
2. `SolvBTC`
    1. `SOLVBTC_POOL_BURNER`:
        1. Can burn unrestricted amounts of `SolvBTC` tokens from any address.
    2. `SOLVBTC_MINTER_ROLE`
        1. Can mint unrestricted amounts of `SolvBTC` tokens to any address.
        2. Can burn unrestricted amounts of `SolvBTC` tokens from itself.
    3. `DEFAULT_ADMIN_ROLE`
        1. Can grant the `SOLVBTC_MINTER_ROLE` to any address, including themselves.
        2. Can revoke the `SOLVBTC_MINTER_ROLE` from any address, including themselves.
    4. `0x55C09707Fd7aFD670e82A62FaeE312903940013E`
        1. Can reinitialize the `SolvBTC` contract and transfer all Wrapped SFT previously held in the contract to a provided address.
        2. Is granted the `DEFAULT_ADMIN_ROLE`.
3. `SolvBTCYieldToken`
    1. `owner`
        1. Can update the `_oracle` address.
4. `SolvBTCYieldTokenOracleForSFT`
    1. `owner`
        1. Can update a given `erc20` tokens SFT oracle configuration.
5. `SolvBTCMultiAssetPool`
    1. `admin`
        1. Can add a registered SFT slot for deposit.
        2. Can remove a registered SFT slot, preventing deposit. Note that this is only executable once the `SolvBTCMultiAssetPool` contract has a zero balance of the SFT slot.
        3. Can claim all empty, unregistered SFTs from the `SolvBTCMultiAssetPool` contract.
        4. Can prevent users from withdrawing.
6. `SftWrapRouter`
    1. `admin`
        1. Can update the `openFundMarket` contract address.
        2. Can update the `sftWrappedTokenFactory` contract address.
        3. Can update the `solvBTCMultiAssetPool` contract address.

Findings

SOLV-1 Attacker Can Drain Value From Contract

Medium | Mitigated

Update

The client provided the following explanation:

```
Accept the recommendation: currently don't allow multiple assets to be stored in single SolvBTCMultiAssetPool on any chains, until next upgrade. Further explanation: currently we do have a 'natural' delay between staking and unstaking so that an attacker cannot drain the maximum value in one transaction, this is because the withdraw/unstaking transaction is approved by a multi-sig wallet hence cannot be done within one transaction. However this mechanic cannot be seen directly from the contract code and hence somehow centralized, so that we are going to upgrade it to a new mechanic. So for now, we still accept the recommendation, not to allow multiple assets to be stored in single SolvBTCMultiAssetPool on any chains, until we upgrade the mechanic to the new version.
```

File(s) affected: `SolvBTCMultiAssetPool.sol`, `SftWrapRouter.sol`

Description: The asset pools assume that different slots of a specific semi-fungible token contract maintain an exchange rate of 1:1. However, if any asset/slot depegs, an attacker can drain the respective asset pool of value. This effect is aggravated since there are only implicit delays between staking and unstaking.

Particularly, wBTC is not a safe asset since it frequently depegs as it implements a delay for unwrapping.

Exploit Scenario:
Assume two currencies xBTC and yBTC that are allowed in `SolvBTCMultiAssetPool` with a price relation on the market of 1.1xBTC = 1yBTC:

1. The attacker has 10 yBTC, which is more valuable than xBTC.
2. The attacker swaps 10 yBTC for 11 xBTC based on the current market rates.
3. The attacker calls `deposit()` on the `SolvBTCMultiAssetPool` contract (to save on gas fees) using 11 xBTC, receiving 11 `solvBTC`.
4. The attacker calls `withdraw()` using 11 `solvBTC`, receiving 11 yBTC.
5. The attacker has extracted a value of 1 BTC from the `SolvBTCMultiAssetPool`.
6. The attacker can repeat this process until no yBTC is left in the contract.

Recommendation: Do not allow slots of assets that can depeg in the same asset pool. Even for a potentially unlikely event of depeg, we recommend implementing a delay between staking and unstaking such that an attacker cannot drain the maximum value in one transaction.

SOLV-2 Oracle May Return Outdated Nav Information

Low | Acknowledged

Update
The client provided the following explanation:

```
No modification is necessary. The NAV time is only intended to be used in display purpose, the NAV is not guaranteed to update in certain time cycle by design, an earlier time does not mean the value is outdated。
```

File(s) affected: `SolvBTCYieldTokenOracleForSFT.sol`, `SolvBTCYieldToken.sol`

Description: The `getNav()` function of `SolvBTCYieldTokenOracleForSFT` returns the latest NAV of the specified ERC-20 token but ignores the latest set NAV time. Without validating that the returned NAV time is within a reasonable range, there may be risks of using outdated information. An outdated NAV value may affect the conversion rate between shares and assets of a `SolvBTCYieldToken`.

Recommendation: Consider adding a check on the latest set NAV time to prevent using outdated information.

SOLV-3 Inability to Stake Non-Legacy SFT Slots

Low | Fixed

Update
Addressed in: `71b230f7def3e87b6ac863ae20446dc260ffde51`.
The client provided the following explanation:

```
Keep SftWrapRouter.sol in version V1, and add SolvBTCRouter.sol. SftWrapRouter.sol no longer needs to be compatible with the old version, so SftWrapRouter will no longer need to call functions related to SftWrappedTokenFactory, and this issue is resolved.
  The issue was fixed in  SolvBTCRouter.sol
```

File(s) affected: `SftWrapRouter.sol`

Description: When executing `stake()` in the `SftWrapRouter` contract, the SFT wrapped token registered for the SFT slot in the `SftWrappedTokenFactory` contract is obtained via a call to `SftWrappedTokenFactory.sftWrappedTokens()`. If the returned `sftWrappedToken` address is the zero address, execution will fail, and staking cannot occur for the given SFT slot. This check prevents users

from staking SFT slots that are only registered in the `SolvBTVMultiAssetPool` and not registered in `SftWrappedTokenFactory`. Further, the function `_stakeSft()`, which is executed after the `SftWrapRouter` contract receives the SFT, only deposits into the legacy `SftWrappedToken` contract if the `SolvBTCMultiAssetPool` contract disallows the SFT slot. Therefore, the zero address check in `stake()` will prevent non-legacy-supported SFT slots from being staked.

Recommendation: Remove the `sftWrappedToken` zero address check in the `stake()` function.

SOLV-4 Inability to Restrict SFT Slot Deposits

Low Fixed

Update

Addressed in: `44dc8c8a76c4132ec8458f6ec7edd1f815fdd2d0`.
The client provided the following explanation:

```
Modify the logic of removeSftSlotOnlyAdmin, no longer check whether the balance is 0, and change the function name to changeSftSlotAllowedOnlyAdmin. Split the allowed flag in SftSlotInfo into allowDeposit and allowWithdraw, the corresponding flags can be modified through changeSftSlotAllowedOnlyAdmin. Delete the sweep function. Modify the SFT implementation to allow transferring tokenId with a value of 0.
```

File(s) affected: `SolvBTCMultiAssetPool.sol`

Description: When the contract admin attempts to remove an SFT slot from the `SolvBTCMultiAssetPool`, the pool contract must have a zero balance of the given SFT slot before removal. Therefore, if even one user fails to withdraw their SFT, the admin will be unable to remove the slot, which allows other users to deposit the SFT slot continually into the contract.

Recommendation: Consider refactoring the `SolvBTCMultiAssetPool` contract so that the admin can prevent further deposits of a given SFT slot regardless of the pool contract's SFT balance. Then, users can still withdraw their previously deposited SFT while not being able to deposit more SFT.

SOLV-5 Unimplemented Interfaces in `SolvBTCMultiAssetPool`

Informational Acknowledged

Update

The client provided the following explanation:

```
No modification is necessary.  The SftWrapRouter is designed to receive assets from users, accepting ERC-3525 token deposits directly for maximum convenience. This allows users to deposit without the need for balance approvals to the router. The SolvBTCMultiAssetPool, on the other hand, serves as a container for assets rather than an interface for end users. Because during the upgrade SftWrapperToken.sol needs to transfer the 3525 SFT's Id to the Pool, the handling of OnERC3525Receiver cannot be added to SolvBTCMultiAssetPool. For safety, the OnERC3525Receiver function will be added and reverted in the next version.
```

File(s) affected: `SolvBTCMultiAssetPool.sol`

Description: According to the protocol design, the `SolvBTCMultiAssetPool` contract may receive ERC-3525 tokens. Therefore, it is best practice to implement the ERC-165 interface and define the `onERC721Received()` and `onERC3525Received()` functions to return the corresponding magic values.

Some SFT implementations, such as `OpenFundShareDelegate`, allow the ERC-3525 receiver to revert on the ERC-165 `supportsInterface()` call and consider the receiver capable of receiving ERC-3525 tokens. However, this behavior is not guaranteed in other SFT implementations.

Recommendation: Consider explicitly implementing the ERC-165, `IERC721Receiver`, and `IERC3525Receiver` interface in the `SolvBTCMultiAssetPool` contract for better interoperability with ERC-3525 tokens.

SOLV-6 Disconnected Upgrade Process

Informational Fixed

Update

Addressed in: `3fae4977fe67cee11e3d6bd29f53dec71bb239ea`.
The client provided the following explanation:

```
Modify the setImplementation function to simultaneously execute deployBeacon or upgradeBeacon when setting the implementation. Since the factory contract cannot be upgraded, the content of the old
```

factory needs to be imported into the newly deployed factory, thus adding relevant import functions. The existing factory contract remains unchanged, and a new SolvBTCFactory.sol is added.

File(s) affected: `SftWrappedTokenFactory.sol`

Description: The `admin` of the `SftWrappedTokenFactory` contract can upgrade the implementation contract for a given `productType_` by calling `setImplementation()` followed by a call to `upgradeBeacon()`. However, if `upgradeBeacon()` is not called, the `productTypes` mapping in the `SftWrappedTokenFactory` contract will have a mismatched `implementation` address than what is set in the beacon contract.

Recommendation: Consider adding an `implementation` parameter to `deployBeacon()` and `upgradeBeacon()` so that the admin can set the beacon's implementation contract in one step. If implemented, `setImplementation()` could be removed.

SOLV-7 Potential Mismatch in `beacon` Contract

Low | Acknowledged

Update

The client provided the following explanation:

In the SolvBTCFactory.sol contract, the address of the beacon contract cannot be obtained, so verification cannot be performed within the contract. We will conduct strict verification in the deployment script to avoid errors. Additionally, we will change the permission for importing the ProductProxy from the governor to the admin to reduce the risk of errors.

File(s) affected: `SolvBTCFactory.sol`

Description: When calling `importProductProxy()`, the `productTypes[productType_].beacon` may be different from the beacon address used by the imported proxy. Therefore, the imported proxy may not upgrade since it is pointing to a different beacon.

Recommendation: To ensure consistency in beacon addresses, verify in `importProductProxy()` that `productTypes[productType_].beacon` is equal to the `_beacon` address of the `proxy_` contract.

Auditor Suggestions

SOLV-S8 Restrict Solidity Pragma to Support Custom Errors

Fixed

Update

Addressed in: `d68dcb3853577b319bbc78bb4d113f9f955a4278`.
The client provided the following explanation:

```
Modified as recommended.
  Pragma fixed to  0.8.20 .
```

File(s) affected: `SolvBTC.sol`

Description: The `SolvBTC` uses custom errors in the `onERC3525Received()` and `onERC721Received()` functions. Therefore, the solidity pragma should be at least `0.8.4` to support the syntax of custom errors.

Recommendation: Consider modifying the solidity pragma to require at least the 0.8.4 version or above. Also, it is best practice to lock the pragma to a fixed version to prevent using a different solidity version between testing and production.

SOLV-S9 Use 2-Step Transfer for Critical Roles

Fixed

Update

Addressed in: `2ca714aac9df6dcb3d9511d47bf1bd6061681edd`.
The client provided the following explanation:

```
Modified as recommended.
```

File(s) affected: `SolvBTCYieldTokenOracleForSFT.sol`

Description: The `SolvBTCYieldTokenOracleForSFT` contract inherits from the `OwnableUpgradeable` contract, which allows the owner to transfer ownership to an arbitrary contract. It is best practice to use a two-step transfer to prevent ownership from being transferred to an uncontrollable address.

Recommendation: Consider implementing a 2-step transfer method or use `AdminControl` or `Ownable2StepUpgradeable` instead.

SOLV-S10 Missing Input Validation

Mitigated

Alert

The issues introduced during the fix review were not resolved.

Update

The initial recommendations have been addressed in: `c3743a1d2b3deaca9bbc4d19957b3e6a7e490310`.
The client provided the following explanation:

```
Modified as recommended.
```

File(s) affected: `SftWrappedTokenFactory.sol`, `SolvBTCMultiAssetPool.sol`

Description:

1. `SftWrappedTokenFactory.constructor()` should validate `governor_` is not `address(0)`.
2. `SftWrappedTokenFactory.setImplementation()` should validate `implementation_` is not `address(0)`.
3. `SolvBTCMultiAssetPool.initialize()` should validate `admin_` is not `address(0)`.

Fix Review Additions:

1. `SolvBTCMultiAssetPool.addSftSlotOnlyAdmin()` should validate that `erc20_` is not the zero address.
2. `SolvBTCFactory.importBeacon()` should validate `productTypes[productType_].beacon` is the zero address before updating.

Recommendation: Consider implementing the checks.

SOLV-S11 Gas Savings

Fixed

Update

Addressed in: `58b366b5cbd4435e9bad00704d337110a602f417`.
The client provided the following explanation:

```
1. No modification is necessary.  SolvBTC does not use Ownable2StepUpgradeable, but SolvBTCYieldToken
   inherits SolvBTC and requires management by the owner, so it cannot be removed.
2. Modified as recommended.
```

File(s) affected: `SolvBTC.sol`, `SolvBTCYieldTokenOracleForSFT.sol`

Description:

1. Remove the unused parent contract `Ownable2StepUpgradeable` from `SolvBTC`.
2. In `SolvBTCYieldTokenOracleForSFT.getNav()` and `SolvBTCYieldTokenOracleForSFT.navDecimals()`, cache `sftOraclesp[erc20]` to avoid multiple `SLOAD` operations.

Recommendation: Consider implementing the gas savings.

SOLV-S12 Gas Overhead

Acknowledged

Update

The client provided the following explanation:

```
No modification is necessary.  Now it is necessary to support the function of transferring to an
existing Sft TokenId when users unstake. During the implementation of this function, holdingValueSftId
needs to be used as an intermediary.
```

File(s) affected: `SftWrapRouter.sol`

Description: The usage of a `holdingValueSftIds` in the `SftWrapRouter` may be redundant and cause a gas overhead.

Since the contract never stores any value beyond a single transaction, the `holdingValueSftId` is not necessarily needed. It only prevents the existence of empty SFT IDs in case the user wants to transfer out value to an existing SFT ID. Nevertheless, the usage of a holding value SFT causes the following overhead:

1. When receiving token value through `onERC3525Received()`, a new token ID is created in L148 to prevent the holding SFT from being transferred.
2. Throughout the contract, a distinction needs to be made between a respective `holdingValueSftId` already existing or assigning a new one. This adds unnecessary complexity.
3. For all (SFT, slot) tuples, an empty SFT may exist in the contract between transactions.

Recommendation: Remove `holdingValueSftIds` in the `SftWrapRouter`. Simply forward the entire token that was received or newly generated. This also prevents empty SFT IDs in the contract, but the user would not be able to transfer out only value when unstaking. This removes the overhead of transfering value to the respective `holdingValueSftId`, then creating a new token, and then sending that new token.

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 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 .`

Automated Analysis

Slither

Using slither, we have analyzed 59 contracts with 94 detectors. Slither identified 116 result(s), which we did not consider relevant for the report.

Test Suite Results

All 5 Testing Suites containing 98 tests could be successfully run.

```
Ran 10 tests for test/SftWrappedTokenFactory.t.sol:SftWrappedTokenFactoryTest
[PASS] test_DeployProduct() (gas: 945507)
[PASS] test_RevertWhenDeployBeaconByNonAdmin() (gas: 49152)
[PASS] test_RevertWhenDeployProductByNonGovernor() (gas: 422035)
[PASS] test_RevertWhenRemoveProductByNonGovernor() (gas: 908007)
[PASS] test_RevertWhenSetImplementationByNonAdmin() (gas: 17922)
[PASS] test_RevertWhenTransferBeaconOwnershipByNonAdmin() (gas: 417730)
```

```
[PASS] test_RevertWhenUpgradeBeaconByNonAdmin() (gas: 4130249)
[PASS] test_RoleSet() (gas: 15663)
[PASS] test_TransferBeaconOwnership() (gas: 430800)
[PASS] test_UpgradeBeacon() (gas: 4646058)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 337.27ms (94.45ms CPU time)

Ran 19 tests for test/SftWrappedToken.t.sol:SftWrappedTokenTest
[PASS] test_BurnWithGivenSftId() (gas: 340121)
[PASS] test_BurnWithoutGivenSftIdWhenHoldingEmptySftIdsIsBlank() (gas: 583555)
[PASS] test_BurnWithoutGivenSftIdWhenHoldingEmptySftIdsIsNotBlank() (gas: 768870)
[PASS] test_InitialMintWithAllValue() (gas: 280509)
[PASS] test_InitialMintWithPartialValue() (gas: 512788)
[PASS] test_InitialTransferWithId() (gas: 260195)
[PASS] test_InitialTransferWithValue() (gas: 479909)
[PASS] test_NonInitialMintWithAllValue() (gas: 482311)
[PASS] test_NonInitialMintWithPartialValue() (gas: 358131)
[PASS] test_NonInitialTransferWithId() (gas: 461911)
[PASS] test_NonInitialTransferWithValue() (gas: 327818)
[PASS] test_RevertWhenBurnWithSftIdNotOwned() (gas: 265133)
[PASS] test_RevertWhenBurnWithSftIdOfInvalidSlot() (gas: 267084)
[PASS] test_RevertWhenBurnWithZeroAmount() (gas: 249719)
[PASS] test_RevertWhenDirectlyCallOnERC3525ReceivedFunction() (gas: 49603)
[PASS] test_RevertWhenDirectlyCallOnERC721ReceivedFunction() (gas: 25385)
[PASS] test_RevertWhenTransferIdOfInvalidSlot() (gas: 172003)
[PASS] test_RevertWhenTransferValueOfInvalidSlot() (gas: 387107)
[PASS] test_RevertWhenTransferValueToNonHoldingValueSftId() (gas: 980156)
Suite result: ok. 19 passed; 0 failed; 0 skipped; finished in 430.82ms (264.91ms CPU time)

Ran 29 tests for test/SolvBTCMultiAssetPool.t.sol:SolvBTCMultiAssetPoolTest
[PASS] test_AddSftSlot() (gas: 249902)
[PASS] test_FullDepositWhenHoldingValueSftIdIsNotZero() (gas: 678383)
[PASS] test_FullDepositWhenHoldingValueSftIdIsZero() (gas: 906199)
[PASS] test_InitialStatus() (gas: 31559)
[PASS] test_PartialDepositWhenHoldingValueSftIdIsNotZero() (gas: 570748)
[PASS] test_PartialDepositWhenHoldingValueSftIdIsZero() (gas: 1268856)
[PASS] test_RemoveSftSlot() (gas: 191203)
[PASS] test_RevertWhenAddExistedSftSlot() (gas: 201460)
[PASS] test_RevertWhenAddSftSlotByNonAdmin() (gas: 20439)
[PASS] test_RevertWhenAddSftSlotWithSftIdNotOwned() (gas: 64996)
[PASS] test_RevertWhenAddSftSlotWithSftIdOfMismatchedSlot() (gas: 58956)
[PASS] test_RevertWhenDecimalsNotMatched() (gas: 49483)
[PASS] test_RevertWhenDepositToInvalidSftSlot() (gas: 87962)
[PASS] test_RevertWhenDepositValueExceedsSftIdBalance() (gas: 261336)
[PASS] test_RevertWhenDepositValueIsZero() (gas: 241082)
[PASS] test_RevertWhenDepositWithSftIdNotOwned() (gas: 251919)
[PASS] test_RevertWhenRemoveInexistentSftSlot() (gas: 20104)
[PASS] test_RevertWhenRemoveNonEmptySftSlot() (gas: 207065)
[PASS] test_RevertWhenRemoveSftSlotByNonAdmin() (gas: 200269)
[PASS] test_RevertWhenWithdrawButPoolHoldingValueNotEnough() (gas: 248862)
[PASS] test_RevertWhenWithdrawButPoolHoldingValueSftIdIsZero() (gas: 240067)
[PASS] test_RevertWhenWithdrawToInvalidSftSlot() (gas: 68688)
[PASS] test_RevertWhenWithdrawToSftIdNotOwned() (gas: 282981)
[PASS] test_RevertWhenWithdrawValueExceedsBalance() (gas: 258871)
[PASS] test_RevertWhenWithdrawValueIsZero() (gas: 239391)
[PASS] test_RevertWhenWithdrawWithMismatchedSpecifiedSlot() (gas: 478884)
[PASS] test_WithdrawForThoseWhoDepositBeforeUpgrade() (gas: 662588)
[PASS] test_WithdrawSlotAAfterDepositSlotA() (gas: 614061)
[PASS] test_WithdrawSlotBAfterDepositSlotA() (gas: 1042841)
Suite result: ok. 29 passed; 0 failed; 0 skipped; finished in 448.22ms (331.50ms CPU time)

Ran 22 tests for test/SolvBTC.t.sol:SolvBTCTest
[PASS] test_AccessControlInitialStatus() (gas: 6729723)
[PASS] test_BurnByAnotherMinter() (gas: 6763008)
[PASS] test_BurnBySolvBTCMultiAssetPool() (gas: 6731866)
[PASS] test_ERC165() (gas: 6712527)
[PASS] test_GrantAdminRole() (gas: 6745297)
[PASS] test_GrantMinterRole() (gas: 6746478)
[PASS] test_MintByAnotherMinter() (gas: 6763051)
[PASS] test_MintBySolvBTCMultiAssetPool() (gas: 6731817)
[PASS] test_OwnershipInitialStatus() (gas: 6711885)
[PASS] test_RenounceAdminRole() (gas: 6732999)
[PASS] test_RevertWhenAcceptOwnershipByNonPendingAdmin() (gas: 6730751)
```

```
[PASS] test_RevertWhenBurnByNonMinter() (gas: 6717398)
[PASS] test_RevertWhenCallInitializeRepeatedly() (gas: 6707110)
[PASS] test_RevertWhenCallInitializeV2Repeatedly() (gas: 6705598)
[PASS] test_RevertWhenGrantRoleByNonAdminRole() (gas: 6745484)
[PASS] test_RevertWhenMintByNonMinter() (gas: 6717374)
[PASS] test_RevertWhenReceivingERC3525() (gas: 6996272)
[PASS] test_RevertWhenReceivingERC721() (gas: 6789317)
[PASS] test_RevertWhenTransferOwnershipByNonAdmin() (gas: 6706783)
[PASS] test_RevokeAdminRole() (gas: 6733741)
[PASS] test_SolvBTCStatusAfterUpgrade() (gas: 6776571)
[PASS] test_TransferOwnership() (gas: 6722244)
Suite result: ok. 22 passed; 0 failed; 0 skipped; finished in 448.53ms (1.31s CPU time)

Ran 18 tests for test/SftWrapRouter.t.sol:SftWrapRouterTest
[PASS] test_CancelRedemption() (gas: 2266264)
[PASS] test_CreateRedemption() (gas: 1577967)
[PASS] test_CreateSubscription() (gas: 773371)
[PASS] test_FirstStakeWithAllValue() (gas: 427890)
[PASS] test_FirstStakeWithPartialValue() (gas: 957263)
[PASS] test_NonFirstStakeWithAllValue() (gas: 786554)
[PASS] test_NonFirstStakeWithPartialValue() (gas: 1443380)
[PASS] test_RouterInitialStatus() (gas: 35434)
[PASS] test_SWT_OnERC3525Received_FirstStake() (gas: 898273)
[PASS] test_SWT_OnERC3525Received_NotFirstStake() (gas: 1383528)
[PASS] test_SWT_OnERC721Received_FirstStake() (gas: 412624)
[PASS] test_SWT_OnERC721Received_NotFirstStake() (gas: 717377)
[PASS] test_SolvBTC_OnERC3525Received_FirstStake() (gas: 899360)
[PASS] test_SolvBTC_OnERC3525Received_NotFirstStake() (gas: 1400536)
[PASS] test_SolvBTC_OnERC721Received_FirstStake() (gas: 449983)
[PASS] test_SolvBTC_OnERC721Received_NotFirstStake() (gas: 733127)
[PASS] test_UnstakeWhenGivenSftId() (gas: 810470)
[PASS] test_UnstakeWhenNotGivenSftId() (gas: 1003708)
Suite result: ok. 18 passed; 0 failed; 0 skipped; finished in 448.43ms (503.39ms CPU time)

Ran 5 test suites in 588.03ms (2.11s CPU time): 98 tests passed, 0 failed, 0 skipped (98 total tests)
```

Code Coverage

The test suite only achieves an average statement coverage of 77% and a branch coverage of 59%. We highly recommend increasing these metrics to 100% to ensure the contracts work as intended and prevent bugs. However, while a thorough testing suite is necessary, it is not a guarantee for security. Consider applying mutation testing beyond improving the testing suite.

File	% Lines	% Statements	% Branches	% Funcs
contracts/SftWrapRouter.sol	80.39% (**123/**153)	83.42% (**166/**199)	50.00% (**38/**76)	61.11% (**11/**18)
contracts/SftWrappedToken.sol	76.00% (**57/**75)	70.21% (**66/**94)	69.64% (**39/**56)	56.25% (**9/**16)
contracts/SftWrappedTokenFactory.sol	89.74% (**35/**39)	90.24% (**37/**41)	45.45% (**10/**22)	100.00% (**10/**10)
contracts/SolvBTC.sol	79.49% (**31/**39)	75.00% (**36/**48)	50.00% (**7/**14)	83.33% (**10/**12)
contracts/SolvBTCMultiAssetPool.sol	87.84% (**65/**74)	84.34% (**70/**83)	85.42% (**41/**48)	90.91% (**10/**11)
contracts/SolvBTCYieldToken.sol	0.00% (**0/**13)	0.00% (**0/**21)	0.00% (**0/**2)	0.00% (**0/**7)
contracts/utils/ERC20TransferHelper.sol	73.33% (**11/**15)	73.68% (**14/**19)	27.27% (**6/**22)	100.00% (**3/**3)

File	% Lines	% Statements	% Branches	% Funcs
contracts/utils/ERC3525TransferHelper.sol	70.00% (**14/**20)	70.97% (**22/**31)	100.00% (**0/**0)	70.00% (**7/**10)
Total	78.50% (**336/**428)	76.68% (**411/**536)	58.75% (**141/**240)	68.97% (**60/**87)

Changelog

- 2024-07-18 - Initial report
- 2024-08-01 - Fix Review 1
- 2024-08-09 - Fix Review 2
- 2024-08-15 - Fix Review 3

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

FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and and may not be represented as such. No third party is entitled to rely on the report in any any way, including for the purpose of making any decisions to buy or sell a product, product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or or any open source or third-party software, code, libraries, materials, or information to, to, called by, referenced by or accessible through the report, its content, or any related related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any any transaction between you and any third party. 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.