# yAudit Yearn - STB Review

**Review Resources:**

- codebase
- architecture diagram

**Auditors:**

- HHK
- Panda

## Table of Contents

## Review Summary

**Yearn - STB**

Yearn - STB provides a bridge for L1 assets to polygon L2s. Its differentiating factor is that the assets staying on the L1 are used as deposits in Yearn vaults, generating yield that could later be redistributed.

The contracts of the Yearn - STB Repo were reviewed over 7 days. The code review was performed by 2 auditors between May 14 and May 22, 2024. The repository was under active development during the review, but the review was limited to the latest commit at the start of the review. This was commit 29cf9cd2586f8460eed1967106f481cecf4ff120 for the Yearn - STB repo.

## Scope

The scope of the review consisted of the following contracts at the specific commit:

- RoleManager.sol
- YearnL1Escrow.sol
- L1Deployer.sol
- l2Deployer.sol

After the findings were presented to the Yearn - STB team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, Yearn - STB and users of the contracts agree to use the code at their own risk.

## Code Evaluation Matrix

| Category | Mark | Description |
|---|---|---|
| Access Control | Average | Important functions are protected, although the different roles over each contract sometimes make it unclear. |
| Mathematics | Good | No complex mathematics is involved. |
| Complexity | Average | The contracts logic is pretty simple, but the multichain nature of the protocol and properties of the STB bridge increase the complexity. |
| Libraries | Low | The protocol is built on top of the zkevm-stb library; it's new and had no usage before yearn-stb to be found. It has been audited once and should be used cautiously. |
| Decentralization | Low | Multiple admins are involved throughout the life cycle of the protocol. Whether it's for deployment, yield strategies, or bridge ownership. Admins can take control of all the funds. |
| Code stability | Good | No commits or changes were introduced during the audit. |
| Documentation | Average | The architecture diagram describes what the protocol is trying to achieve, but there is currently no documentation and the Polygon STB's documentation is not always clear. |
| Monitoring | Good | Important functions emit events. |

| Category | Mark | Description |
|---|---|---|
| Testing and verification | Average | Good test coverage is achieved throughout the codebase with a few exceptions. Testing cross-chain is always difficult. |

## Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
  - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
  - Findings that can improve the gas efficiency of the contracts.
- Informational
  - Findings including recommendations and best practices.

---

## Critical Findings

None.

## High Findings

None.

## Medium Findings

None.

## Low Findings

### 1. Low - Some tokens will not be compatible

Some tokens will not be compatible:

- Some tokens (e.g. UNI, COMP) revert if the value passed to approve or transfer is larger than uint96.

#### Technical Details

```
File: src/L1YearnEscrow.sol
100 |    originTokenAddress().forceApprove(_vaultAddress, 2 ** 256 - 1);
247 |    originToken.forceApprove(_vaultAddress, 2 ** 256 - 1);
```

L1YearnEscrow.sol#L100

- Some tokens have a fee on transfer which will result in more tokens bridged to L2 than received by the L1 escrow.

```
function bridgeToken(address destinationAddress, uint256 amount, bool
forceUpdateGlobalExitRoot) external {
        require(destinationAddress != address(0), "TokenWrapped::PolygonBridgeERC20Base:
Zero Address");
        _receiveTokens(amount);

        // Encode message data
        bytes memory messageData = abi.encode(destinationAddress, amount);

        // Send message data through the bridge
        _bridgeMessage(messageData, forceUpdateGlobalExitRoot);

        emit BridgeTokens(destinationAddress, amount);
    }
```

PolygonERC20BridgeBaseUpgradeable

#### Impact

Low. Only a small number of tokens are affected.

#### Recommendation

- You could add an optional parameter to the `initialize` and `updateVault` to pass a max approval value.
- Since the `bridgeToken()` function can't be overridden, you could check the balance actually received by the L1 escrow in `_receiveTokens()` and revert if it differs from the `amount`.

#### Developer Response

Acknowledged.

As discussed in Discord as long as the approval to UNI/COMP is max uint256 the approvals work.

Weird tokens like fee on transfer tokens would not work with the vaults as well so should not be used at all with this setup.

### 2. Low - Inconsistent result when escrow is already deployed

#### Technical Details

In the `newEscrow()` function if an escrow is already deployed then the function will revert with `AlreadyDeployed()`.

But in `newCustomVault()` the function will not revert and just ignore the deployment.

This can lead to a rollup admin thinking they deployed an escrow with a custom vault when, in fact, the escrow was already deployed and is not using this new custom vault.

#### Impact

Low.

#### Recommendation

Consider always reverting with `AlreadyDeployed()`.

#### Developer Response

Partially fixed.

Updated the two newCustomVault functions slightly and added some more comments in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

The first version will always deploy a vault (which will revert if one already has been deployed). But only deploy an escrow if one does not exist. And expects the admin to update their existing escrow.

The second one will always deploy and escrow (which will revert if one already exists).

This gives the admins full control to add/change custom vaults but will make sure the functions are actually doing something correct.

### 3. Low - Previous `keeper` might not be removed in `updateKeeper()`

#### Technical Details

In the function, `updateKeeper()` the previous `keeper` is only removed if it was the default `keeper`.

This means that if the `keeper` was updated once from default to a new one, it would properly remove the default one and add the new one. But if the `MANAGEMENT` tried to update it a second time, it wouldn't remove the previous `keeper`, and the vault would end up with the new and old `keeper`.

This can be an issue if a `keeper` is compromised and the `MANAGEMENT` uses this function to set a new one. The compromised `keeper` will still be able to interact with the vault.

#### Impact

Low. `removeRoles()` can be called to remove the previous `keeper`.

#### Recommendation

Always remove the previous `keeper` and not just when it was using the default one.

#### Developer Response

acknowledged. As stated removeRoles can be used.

## Gas Saving Findings

### 1. Gas - Add `unchecked {}` for subtractions where operands are certain not to underflow

The following operations are math-safe.

#### Technical Details

```
File: L1YearnEscrow.sol

184 | uint256 shares = _vault.convertToShares(amount - maxWithdraw);

254 | balance - _minimumBuffer,

288 | $.vaultAddress.deposit(balance - _minimumBuffer, address(this));

291 | uint256 diff = _minimumBuffer - balance;
```

L1YearnEscrow.sol#L184, L1YearnEscrow.sol#L254, L1YearnEscrow.sol#L288, L1YearnEscrow.sol#L291

#### Impact

Gas.

#### Recommendation

Use `unchecked` blocks.

#### Developer Response

Fixed in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

## 2. Gas - Use cached variable instead of loading storage

#### Technical Details

- In the `initialize()` function of the L1YearnEscrow use `_originTokenAddress` instead of `originTokenAddress()` to save gas.
- In the `_receiveTokens()` function of the L1YearnEscrow cache `originTokenAddress()` on first storage load to not load twice.

#### Impact

Gas.

#### Recommendation

Use cached variables instead of loading storage.

#### Developer Response

Fixed in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`

## Informational Findings

### 1. Informational - State variables not capped at reasonable values

Consider adding minimum/maximum value checks to ensure that the state variables below can never be used with extreme values.

#### Technical Details

```
File: RoleManager.sol


526 | defaultProfitMaxUnlock = _newDefaultProfitMaxUnlock;
```

RoleManager.sol#L526

#### Impact

Informational.

#### Recommendation

Add min/max value check.

#### Developer Response

Fixed in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

Added the same max as the vault enforces.

### 2. Informational - `public` functions not called by the contract should be declared `external` instead

For clarity.

### Technical Details

```
File: L1Deployer.sol

345 | function getL2Deployer(

367 | function getEscrowManager(
```

L1Deployer.sol#L345, L1Deployer.sol#L367

```
File: L1YearnEscrow.sol

53 | function vaultAddress() public view returns (address) {

58 | function minimumBuffer() public view returns (uint256) {

78 | function initialize(
```

L1YearnEscrow.sol#L53, L1YearnEscrow.sol#L58, L1YearnEscrow.sol#L78

```
File: RoleManager.sol

571 | function getVault(address _asset) public view virtual returns (address) {

601 | function isVaultsRoleManager(
```

RoleManager.sol#L571, RoleManager.sol#L601

#### Impact

Informational.

#### Recommendation

Change the visibility to external.

#### Developer Response

Partially fixed.

Updated for the Role manager and the L1Deployer in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

Not changed for the `L1YearnEscrow` in case for some reason the contract is ever inherited to be built on top of. Will want getter functions for the custom stored variables just like we use with `originTokenAddress()`.

## 3. Informational - Avoiding unnecessary integer variable initialization

In Solidity, variables are automatically initialized to zero. Therefore, explicitly initializing variables to zero is not mandatory and can be skipped to optimize gas costs.

#### Technical Details

```
File: RoleManager.sol

46 | uint32 internal constant ORIGIN_NETWORK_ID = 0;

473 | for (uint256 i = 0; i < _vaults.length; ++i) {
```

RoleManager.sol#L46, RoleManager.sol#L473

#### Impact

Informational.

#### Recommendation

Avoid unnecessary integer variable initialization.

#### Developer Response

Partially fixed.

Updated for the for loop in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

Kept the constant as its will only happen once and is preferred for clarity to keep it as is.

## 4. Informational - Refactor duplicated require()/revert() checks into a modifier or function

Refactoring duplicated require()/revert() checks into a modifier or function is recommended. By doing so, the compiler will inline the function, eliminating the need for JUMP instructions typically associated with functions.

### Technical Details

```
File: L1Deployer.sol
166 | require(_l2Deployer != address(0), "ZERO ADDRESS");
209 | require(_l2Deployer != address(0), "ZERO ADDRESS");
```

L1Deployer.sol#L166, L1Deployer.sol#L209

```
File: RoleManager.sol
383 | require(vaultConfig[_vault].asset != address(0), "vault not added");
411 | require(vaultConfig[_vault].asset != address(0), "vault not added");
476 | require(vaultConfig[_vault].asset != address(0), "vault not added");
```

RoleManager.sol#L383, RoleManager.sol#L411, RoleManager.sol#L476

### Impact

Informational.

### Recommendation

Use a modifier.

### Developer Response

Partially fixed.

Updated for the `vault not added` checks in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

Not added for the zero address checks.

## 5. Informational - Missing checks for `address(0)` when assigning values to address state variables

A check to ensure the address(0) is not set is missing.

#### Technical Details

```
File: L1Deployer.sol

176 | _chainConfig[_rollupID].escrowManager = _l1EscrowManager;
```

L1Deployer.sol#L176

```
File: RoleManager.sol

395 | vaultConfig[_vault].debtAllocator = _debtAllocator;

395 | vaultConfig[_vault].debtAllocator = _debtAllocator;
```

RoleManager.sol#L395, RoleManager.sol#L395

#### Impact

Informational.

#### Recommendation

Add a check to make sure the address is not zero.

#### Developer Response

Partially fixed.

Updated for the l1Escrow manager in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

Not added for the debt allocator to allow for the potential of and address(0) debtAllocator to be set.

### 6. Informational - Use `string.concat()` on strings instead of `abi.encodePacked()` for clearer semantic meaning

Starting with version 0.8.12, Solidity has the `string.concat()` function, which allows one to concatenate a list of strings, without extra padding. Using this function rather than `abi.encodePacked()` makes the intended operation clear, leading to less reviewer confusion.

#### Technical Details

```
File: RoleManager.sol

188 | string(abi.encodePacked("-", Strings.toString(_rollupID)));

192 | abi.encodePacked(ERC20(_asset).symbol(), "-STB", _id, " yVault")

196 | abi.encodePacked("stb", ERC20(_asset).symbol(), _id)
```

RoleManager.sol#L188, RoleManager.sol#L192, RoleManager.sol#L196

#### Impact

Informational.

#### Recommendation

Use `string.concat()`.

#### Developer Response

Fixed in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

### 7. Informational - `updateVault` will fail if not enough tokens can be withdrawn

Although the contract enforces `maxWithdraw` checks in `_transferTokens`, it lacks similar validations in `updateVault`, which could potentially withdraw more than the maximum vault. Vault migrations are expected to be coordinated with the vault owner to ensure full balance availability.

#### Technical Details

```
File: L1YearnEscrow.sol
225 |     function updateVault(
226 |         address _vaultAddress
227 |     ) external virtual onlyRole(DEFAULT_ADMIN_ROLE) {
```

L1YearnEscrow.sol#L225-L227

#### Impact

Informational.

#### Recommendation

Add a comment regarding the vault being liquid for the entire withdrawal to the natspec.

#### Developer Response

Fixed, comment added in commit `d8881fb1e43988c85f685c4fa7e2de4cead1d407`.

### 8. Informational - Consider ABA pattern for escrow deployment

#### Technical Details

When calling the `newEscrow()` function the L1 deployer will create the L1 escrow and then send a message to the bridge towards the L2 deployer so it deploys the L2 escrow.

The message to the L2 deployer can take some time to execute or can hypothetically revert. In that case, any user using the L1 escrow to start bridging will have his tokens stuck until the L2 escrow is deployed.

#### Impact

Informational. It seems unlikely that the L2 escrow deployment would be delayed or would revert.

#### Recommendation

Implement ABA pattern on the L1 escrow.

This means not allowing bridging until the L2 escrow is deployed and notified the L1 escrow (L1 -> L2 -> L1).

By having the L2 escrow callback the bridge after deployment to notify the L1 escrow that will then enable the bridging.

#### Developer Response

Acknowledged.

Should not be a concern and the L1 rollup admins can fix any mistakes through the L1 escrow if need be.

## Final remarks

In conclusion, the Yearn - STB contract audit identified various findings, primarily low to informational impact, suggesting optimizations for gas efficiency, clarity, and robustness. Key areas like access control, mathematics, and monitoring were satisfactory, though decentralization and new library risks were noted.