OpenZeppelin | security

Paxos wYBS Token Audit

September 19, 2024

Table of Contents

Summary

Type	DeFi
Timeline	From 2024-08-18 To 2024-08-23
Languages	Solidity, Javascript

Total Issues	19 (17 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	2 (2 resolved)
Low Severity Issues	7 (7 resolved)
Notes & Additional Information	10 (8 resolved)

Scope

We audited the paxos-global/ybs-contract-internal repository at commit 2cf392a.

In scope were the following files:

```
ybs-contracts-internal
└── contracts
    └── wYBS.sol
```

We also considered the `contracts/YBS.sol` contract and the contracts inside the `contracts/lib` directory to the extent of their relation and interaction with the `wYBS` contract.

In addition, we also performed testing and configuration assessment of the deploy script in order to validate that the result of the script deploys correctly.

In scope were the following files:

```
ybs-contracts-internal
└── scripts
    └── wYBS
        └── deploy.js
```

System Overview

The YBS token is a yield-bearing stablecoin by Paxos. It accrues yield for its holders on a daily basis through a rebasing mechanism. In essence, token holders can see their YBS balance growing day by day without any action from their side.

Most DeFi protocols are incompatible with rebasing tokens like YBS because the consistency of their internal mechanisms and accounting depends on a fixed balance assumption. For this reason, Paxos implemented the wYBS token, which is a non-rebasing wrapped version of YBS.

The wYBS token follows the EIP-4626 specification and is implemented using the OpenZeppelin `ERC4626Upgradeable.sol` library. The `wYBS` contract behaves as a YBS pool. Users are able to deposit YBS tokens and receive wYBS tokens in return which act as shares of the pool. The pool's liquidity grows over time due to its rebasing nature and the users are eligible for a portion of it in proportion to their share.

Deployment Scripts Review

In addition to manual review of the codebase, the engagement also incorporated testing and configuration assessment of the deploy scripts.

The deployment script for the `wYBS` contract was reviewed to ensure it adheres to best practices and does not introduce any security vulnerabilities. The script was thoroughly analyzed for potential issues related to contract initialization, role assignment, and overall deployment security. After a careful review, the following key aspects were validated:

1. **Contract Initialization and Constructor Arguments**
    - The constructor/initializer arguments were correctly passed and encoded.
    - The script appropriately sets up the initialization parameters (`contractName`, `contractSymbol`, `YBS_ADDRESS`, `ADMIN_ADDRESS`, `PAUSER`, and `ASSET_PROTECTOR`) ensuring that the contract is configured as intended upon deployment.

- Variables are in the correct order to match the expected constructor or initializer function in the contract.
- There are no invalid values being used during initialization, which could lead to unintended behavior or security vulnerabilities. Except for the `contractName` and `contractSymbol`, which are using mocked values.

2. **Contract Deployment**

   - The script deploys the contract using the `deployProxy` function, ensuring that the proxy contract is initialized in the same transaction as its deployment. This mitigates risks associated with uninitialized proxies and front-running.
   - The script utilizes OpenZeppelin's upgradeable proxy pattern (UUPS) for deploying the `wYBS` contract.
   - The contract is confirmed to be a UUPS proxy and is being initialized correctly.

3. **Privileged Roles Assignment**

   - Privileged roles such as `ADMIN_ADDRESS`, `PAUSER`, and `ASSET_PROTECTOR` are set via environment variables. Due to this, the correct assignment of these roles was not directly verified within the script review. It is assumed that the client will use the correct and intended values for these roles, and we consider that this is being handled securely.

4. **Logging and Transparency**

   - The script includes logging of critical information such as the deployer address, account balance, transaction hash, proxy contract address, and implementation contract address.
   - Addresses are logged correctly, allowing for clear verification of deployment details.

5. **Use of Secure Libraries and Practices**

   - The script uses the `hardhat` framework in combination with OpenZeppelin's `upgrades` library, which are widely respected and audited tools in the Ethereum development community.
   - Proper use of environment variables (`process.env`) ensures sensitive information is not hardcoded in the script, reducing exposure risk.
   - There are no hardcoded secrets or keys in the code, which mitigates the risk of accidental exposure.

6. **Code Quality and Accuracy**
    - The script was reviewed for typos or syntax errors, and no issues were found. The code is clean and follows standard practices for readability and maintainability.

In addition to validating the appropriate configuration and parameters of the deployment script, our team also identified the following Low-level issue and Notes:

- **Inaccurate Contract Name and Symbol in Deployment Script (L-05)**
- **Missing Environment Variable Validation in Deployment Scripts (N-05)**
- **Inconsistent Contract Usage in Deployment Script (N-09)**

Security Model and Trust Assumptions

There are a number of privileged roles which have significant power over the `wYBS` contract.

- `DEFAULT_ADMIN_ROLE`

The `DEFAULT_ADMIN_ROLE` is able to upgrade the token to another implementation contract. It is initialized to the `admin` address during the contract's initialization and is able to grant or revoke a role to an address. This role can be renounced or assigned to another address following the two-step procedure of the `AccessControlDefaultAdminRules` library. Between these two steps, a minimum time of 3 hours must elapse so that the users are given enough time to inspect any critical changes and react according to their interests. While the default admin can change this time interval needed to transfer the admin rights, the first admin transfer schedule after the change will still be subject to the initial 3-hour delay period.

- `ASSET_PROTECTION_ROLE`

The `ASSET_PROTECTION_ROLE` is able to block/unblock any account in the token contract. A blocked account cannot deposit, redeem, transfer, and receive tokens or make use of approvals. In addition, this role is able to seize the wrapped assets of a blocked address and transfer them to an arbitrary receiver address.

- `PAUSE_ROLE`

The `PAUSE_ROLE` is able to pause/unpause the token contract. When the contract is paused, no action affecting the token's balances or approvals is allowed, except for seizing a blocked address' balance which the `ASSET_PROTECTION_ROLE` is allowed to do.

It is assumed that all the privileged roles are trusted and act in the best interests of the users. The underlying YBS token is also trusted to behave as expected.

Medium Severity

M-01 Impossible to Seize Assets When Protocol is Paused

In the `wYBS` contract, the `seizeAssets` function is currently restricted from execution while the protocol is paused. This limitation arises from the `_beforeTokenTransfer` function, which applies the `whenNotPaused` modifier. Although this functionality is expected to be available during a pause, it is not currently usable in such a situation.

Consider enabling the execution of the `seizeAssets` function even when the protocol is paused.

Update: *Resolved in pull request #53 at commit f81b6b9 and pull request #73 at commit 23b78fd.*

M-02 Front-Running of `transferWithAuthorizationBatch` Causes Complete Batch Revert

The `transferWithAuthorizationBatch` function processes multiple transfers within a single transaction by iterating through multiple arrays of parameters. A third-party observer can detect this transaction in the mempool and front-run it by executing one of the transfers using the observed data.

If the frontrunner successfully executes the transfer, the corresponding `nonce` will be marked as used. When the original batch transaction attempts that transfer, it will fail due to the `AuthorizationAlreadyUsed` error, causing the entire batch to revert. Similarly, when the `cancelAuthorization` function is called in a complex transaction, a third-party observer can front-run with duplicate calldata, forcing the original transaction to revert.

Consider revising the `transferWithAuthorizationBatch` function. For example, one option could be allowing users to execute the entire batch anyway while emitting an event for the failed transfers. However, this approach would break the atomicity of batch execution as some transfers would succeed while others would not.

Update: *Resolved in pull request #58 at commit 2dd6860.*

Low Severity

L-01 Missing Docstrings

Within `wYBS.sol`, multiple instances of missing docstrings were identified:

- The `PAUSE_ROLE` state variable
- The `ASSET_PROTECTION_ROLE` state variable
- The `DOMAIN_SEPARATOR` function

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Resolved in pull request #59 at commit 2b8d33a.*

L-02 Incorrect Implementation of EIP-3009

In the `wYBS` contract, the `_transferWithAuthorization` function differs from the EIP-3009 standard, potentially allowing unauthorized transfers. The function permits a transfer if the current timestamp is equal to or greater than `validAfter`, and on or before `validBefore`. In contrast, EIP-3009 mandates that a transfer is only valid if the current time is strictly after `validAfter` and before `validBefore`. This discrepancy allows the `wYBS` contract to accept transfers that should be invalid under EIP-3009, which could result in unauthorized actions.

Consider changing the conditions in the `_transferWithAuthorization` function to align with the EIP-3009 standard, ensuring that transactions are only valid if the current time is strictly after `validAfter` and before `validBefore`.

Update: *Resolved in pull request #57 at commit 5b64e09.*

L-03 Potential Unauthorized Transfers by `ASSET_PROTECTOR`

The `wYBS` contract grants the `ASSET_PROTECTOR` role the ability to execute the `seizeAssets` function, which burns `wYBS` tokens and withdraws `YBS` tokens to a specified receiver. In addition, the `ASSET_PROTECTOR` is allowed to perform the `transferFrom`, `withdraw`, and `redeem` functions for blocked accounts due to the `if` statement present in the `_spendAllowance` function. This may not align with the intended permissions of the `ASSET_PROTECTOR` role.

Consider reviewing and restricting the `ASSET_PROTECTOR` role's access to the `transferFrom`, `withdraw`, and `redeem` functions to prevent potential unauthorized transfers from blocked accounts.

Update: *Resolved in pull request #67 at commit 853251b and pull request #73 at commit 23b78fd.*

L-04 Possible to Grant Allowance to a Blocked Account

The `wYBS` contract is designed to prevent users from granting approval to blocked accounts. While this restriction is enforced by the `approve` and `increaseAllowance` functions, it is bypassed by the `permit` function, which allows granting of allowances to blocked accounts.

Consider adding a check that `spender` is not present in the blocklist.

Update: *Resolved in pull request #60 at commit 6616537.*

L-05 Inaccurate Contract Name and Symbol in Deployment Script

The deployment scripts for the `wYBS` and `YBS` contracts are currently using mock values for the `contractName` and `contractSymbol` variables.

To ensure accuracy and consistency, consider updating these variables to reflect the actual contract names and symbols corresponding to the `YBS` and `wYBS` contracts.

Update: *Resolved in pull request #61 at commit 2d61d76.*

L-06 Rebase Multiplier Can Be Decreased

The `setRebaseMultipliers` function in the `YBS` contract allows the `REBASE_ADMIN_ROLE` to change the value of the rebase multiplier. Although the rebase multiplier is only supposed to increase, `setRebaseMultipliers` allows arbitrarily defining the `beforeIncrMult` variable. As a result, even a reduced value for the rebase multiplier is allowed to be set.

Consider revising the logic so that rebasing can only ever result in an increase in tokens, if that is desired.

Update: *Resolved in pull request #69 at commit 5b70995 and pull request #74 at commit 8a47763. The team stated:*

> *This function, `setRebaseMultipliers`, was replaced with a safer function, `setNextMultiplier`. Additionally, a new check was added to prevent setting a multiplier with `multIncrTime` in the past. See the pull request's description for details.*

L-07 Inconsistent Block Handling

The `cancelAuthorization` and `_transferWithAuthorization` functions of the `EIP3009` library revert if `msg.sender` is blocked, despite `msg.sender` not being directly involved in the logic. In contrast, the `permit` function of the `EIP2612` library allows this behavior. Currently, this can be bypassed by using another account in functions that don't depend on `msg.sender`.

Consider applying uniform restrictions across all relevant functions if the goal is to enforce a strict policy preventing any interaction from blocked accounts.

Update: *Resolved in pull request #75 at commit 41542d7.*

Notes & Additional Information

N-01 Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice is quite beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. In addition, if the contract incorporates third-party libraries and a bug surfaces in those, it becomes easier for their maintainers to contact the appropriate person about the problem and provide mitigation instructions.

The `wYBS` contract does not provide a security contact.

Consider adding a NatSpec comment containing a security contact above each contract definition. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Update: *Resolved in pull request #62 at commit d83a171.*

N-02 Functions Visibility Overly Permissive

Within `wYBS.sol`, multiple functions with unnecessarily permissive visibility were identified:

- The `seizeAssets` function with `public` visibility can be limited to `external`.
- The `transferFromBatch` function with `public` visibility can be limited to `external`.

To better convey the intended use of functions and to potentially realize some additional gas savings, consider changing a function's visibility to be only as permissive as required.

Update: *Resolved in pull request #63 at commit f6ef5a4.*

N-03 Redundant `v` Check in `recover`

The `recover` function in the `ECRecover` library includes a check to ensure that the `v` component of a signature is 27 or 28. However, this is redundant since the `ecrecover` precompile returns `address(0)` for invalid `v` values.

Consider removing the `v` check, depending on the blockchain environment.

Update: *Resolved in pull request #56 at commit dc7f140.*

N-04 Potential `emit` of Misleading Event

In the `wYBS` contract, the `_blockAccount` and `_unblockAccount` functions do not verify whether the target account's status has changed before emitting the corresponding event. This can lead to confusion, particularly when the `_unblockAccount` function is called and the `AccountUnblocked` event is emitted for an account that was not previously blocked.

Consider adding a check to ensure that the function actually changes the status of the account before emitting the event, thereby preventing the generation of misleading events.

Update: *Acknowledged, not resolved. The Paxos team stated:*

> *Not having this check is intentional in order to reduce the gas cost. This is only called by the `ASSET_PROTECTION_ROLE` which is an internal Paxos operations account.*

N-05 Missing Environment Variable Validation in Deployment Scripts

The deployment scripts for the `wYBS` and `YBS` contracts utilize several environment variables as initialization arguments passed to the upgradeable contract. However, the scripts do not explicitly verify whether these environment variables are set before proceeding.

Consider implementing a validation check to ensure that all the required environment variables are properly defined before initiating the deployment process.

Update: *Resolved in pull request #64 at commit bc41299.*

N-06 EIP-3009 Transactions Lack Protection Against Replay Attacks for Failed Transactions

In the `wYBS` contract, the `EIP3009` functions do not prevent replay attacks when an initial transaction fails. An external observer can reuse the data from the failed transaction to execute it later, potentially when conditions have changed (e.g., after the protocol is unpaused).

Consider revising the logic or making users aware of the problem.

Update: *Resolved in pull request #55 at commit 70bb422. The team stated:*

> *We went with the approach of documenting this issue.*

N-07 Redundant Check in `initialize`

In the `initialize` function of the `wYBS` contract, the check `admin == address(0)` is redundant because this condition is already verified in `__AccessControlDefaultAdminRules_init_unchained`.

Consider removing the redundant check from the `initialize` function.

Update: *Resolved in pull request #65 at commit 78535f8.*

N-08 Redundant Cast

In the `makeDomainSeparator` function of the `EIP712` library, the `chainId` parameter is cast to `bytes32`, while in the domain typehash, it is defined as of type `uint256`.

Although these types are encoded similarly and will produce the same final results, consider removing the redundant cast for improved clarity.

Update: *Resolved in pull request #54 at commit 23fcd18.*

N-09 Inconsistent Contract Usage in Deployment Script

The deployment script for `wYBS` is currently utilizing the `wYBS` contract in the getContractFactory function.

However, to maintain consistency with the `YBS` deployment script which uses the `YBSV1` contract, consider switching the contract reference from `wYBS` to `wYBSV1` in the `wYBS` deployment script.

Update: *Resolved in pull request #66 at commit 202f13f.*

N-10 Unsynchronized YBS and wYBS Blocklists

Both YBS and wYBS tokens define a list of blocked addresses to which any transaction for token transfer or approval is prohibited. Despite the fact that the wYBS token is the wrapped ERC-4626 version of the YBS rebasing token, their blocklists are separately defined in each contract and act independently.

Consider synchronizing the blocklists of the wYBS and YBS tokens on-chain to enhance code consistency and clarity.

Update: *Acknowledged, not resolved. The Paxos team stated:*

> *This approach has been considered, but it is more gas-efficient to have the block lists within the contract compared to making an external call.*

Conclusion

The audited codebase introduces the `wYBS` token, which is a wrapper for the `YBS` yield-bearing stablecoin. `YBS` token holders who wish to interact with DeFi protocols that do not accept rebasing tokens may use the `wYBS` assets instead of `YBS`.

A few medium-severity issues were discovered. In addition, multiple recommendations were made to ensure that the protocol behaves as expected and the overall clarity of the codebase is improved.