SOLIDIFIED

Audit Report for Poseidon DAO - December 9, 2022

Summary

Audit Report prepared by Solidified covering Poseidon DAO's ERC20-ERC1155 token.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on December 2, 2022, and the results are presented here.

Audited Files

The source code has been supplied in the following source code repository:

Repo: https://github.com/Poseidon-DAO/poseidon-token
Commit hash: `520bf1e4c19788285d78ec0db183fb75dd24dd88`

```
contracts
├── ERC1155_PDN.sol
├── ERC20_PDN.sol
└── IERC1155_PDN.sol
```

Intended Behavior

The code base implements an ERC20 token that can be burned to receive ERC1155 NFTs (with a configurable ERC20/ERC1155 ratio).

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	Medium	-
Level of Documentation	High	-
Test Coverage	Medium	-

Issues Found

Solidified found that the Poseidon DAO contracts contain no critical issues, 3 major issues, 3 minor issues, and 6 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	ERC20_PDN.sol: Function deleteVest does not remove vest amounts from ownerLock	Major	Resolved
2	ERC20_PDN.sol: Function runAirdrop inconsistently scales the airdrop amount with _decimals	Major	Resolved
3	ERC20_PDN.sol: The burn functionality is missing ownerLock checks	Major	Resolved
4	Inconsistent decimals for amount parameter	Minor	Resolved
5	ERC20_PDN.sol: Function burnAndReceive does not scale down the msg.sender balance before comparison	Minor	Resolved
6	Vested unclaimed amounts can be deleted	Minor	Resolved
7	ERC20-PDN.sol: Not used event DAOConnectionEvent	Note	Resolved
8	ERC20-PDN.sol: Function burn has a wrong comment	Note	Resolved
9	The project is using SafeMathUpgradeable but it is not needed	Note	Resolved

10	Valid ERC1155 ID 0 not accepted in setERC1155	Note	Resolved
11	airdropVest does not use the constant SECURITY_DELAY	Note	Resolved
12	ERC20_PDN.sol: Function addVest has a _duration parameter that is not a duration in seconds	Note	Resolved

Critical Issues

No critical issues have been found.

Major Issues

1. ERC20_PDN.sol: Function deleteVest does not remove vest amounts from ownerLock

The `ownerLock` storage variable should be subtracted by the amounts from the deleted vests, but it isn't. Failing to do so will leave owner tokens locked forever.

Recommendation

Iterate through the `vestList` for the given address and subtract each amount from `ownerLock`

2. ERC20_PDN.sol: Function runAirdrop inconsistently scales the airdrop amount with _decimals

The `runAirdrop` function scales the amount to transfer with the `_decimals` passed, but it does not do so when subtracting the amount from the `availableOwnerBalance`. This will result in the `availableOwnerBalance` variable holding a much bigger value than the actual balance of the `owner`. The code also does not scale it in the `require` statement for `INSUFFICIENT_OWNER_BALANCE` in the function.

Recommendation

Consistently scale (or do not scale at all) the airdrop amount with the `decimals`

3. ERC20_PDN.sol: The burn functionality is missing ownerLock checks

In both `transfer` and `transferFrom`, before moving any amount from an address, there are checks if that address is the `owner`, and if it is - if he will have at least `ownerLock` balance left after transfer. This is not done neither in `burn` nor in `burnAndReceiveNFT` and can result in the `owner` having a balance that is less than `ownerLock`.

Recommendation

Add the same checks that are in `transfer` and `transferFrom` to `burn` and `burnAndReceiveNFT`

Minor Issues

4. Inconsistent decimals for amount parameter

Some functions that accept an amount multiply this amount by `10 ** _decimals` (`initialize`, `runAirdrop`, `burnAndReceiveNFT`). Other functions (`addVest`, `airdropVest`, all standard ERC20 functions) expect an amount with the token's decimals. This can be confusing and error-prone. Furthermore, dividing a token into smaller units is not possible for the functions that expect an amount with 0 decimals. This may be undesirable when the value of 1 token is large.

Recommendation

Consider using amounts with the token's decimals everywhere.

5. ERC20_PDN.sol: Function burnAndReceive does not scale down the msg.sender balance before comparison

The `burnAndReceive` function is documented to expect to receive an `_amount` argument that is not considering the `decimals` of the token. From the `_amount` argument we calculate the `NFTAmount` variable, which we later compare to the balance of `msg.sender`. This check is incorrect, since the `balanceOf` method always returns the `decimals` scaled balance. It is a minor issue though because even if the check incorrectly passed, the code will fail on the `_burn` call.

Recommendation
Scale down the `msg.sender` balance or accept an `_amount` argument that is considering the `decimals` of the token.

6. Vested unclaimed amounts can be deleted

The function `deleteVest` can be used to delete entries in `vestList` that are already claimable (i.e., where the block number in `expirationDate` is smaller than the current block number), but were not claimed yet by the user. This may be undesirable because vesting usually implies that a user is guaranteed to get the tokens after the expiration date has expired.

Recommendation
Consider only allowing deleting entries that are not expired.

Informational Notes

7. ERC20-PDN.sol: Not used event DAOConnectionEvent

The event `DAOConnectionEvent` is not used anywhere, it is just declared.

Recommendation
Delete unused event `DAOConnectionEvent`

8. ERC20-PDN.sol: Function burn has a wrong comment

The NatSpec of `burn` says it emits `OwnerChangeEvent` which is incorrect, such event is not emitted in the function.

Recommendation

Update the NatSpec to not say it emits `OwnerChangeEvent`

9. The project is using SafeMathUpgradeable but it is not needed

Solidity compiler version 0.8.0 introduced built-in math checks, so there is no need to use a library like `SafeMath` anymore as it will result in gas overhead.

Recommendation

Remove the `SafeMathUpgradeable` library from the project

10. Valid ERC1155 ID 0 not accepted in setERC1155

The ID 0 is a valid ID for an ERC1155 token (also for `ERC1155_PDN`) and there are many token implementations where the IDs start at zero. However, the function setERC1155 requires that the ID is greater than 0. This may be too restrictive.

Recommendation

Consider allowing using the ID 0.

11. airdropVest does not use the constant SECURITY_DELAY

Instead of using the constant `SECURITY_DELAY`, the function `airdropVest` uses the value 5760 directly. This can be error-prone, because the value needs to be changed in multiple cases if the delay is changed at some point in the future.

Recommendation

Consider using the constant instead of the value.

12. ERC20_PDN.sol: Function addVest has a _duration parameter that is not a duration in seconds

---

The `vest` struct has a field `expirationDate` that is not really a date but a value of block height. Because of this, the `_duration` parameter should be an amount of blocks. This is not intuitive and can lead to errors, for example sending a seconds/timestamp value for `_duration` when calling `addVest`, which will result in much longer vesting time.

Recommendation

Rename `expirationDate` to `expirationBlockHeight` and `duration` to `durationInBlocks`