Aligned ERC20 and ClaimableAirdrop

December 2024

Made with ❤ by the following Creed authors: Valentin Quelquejay

Table of Contents

Executive Summary

This report presents the results of our engagement with Aligned to review their ERC20 Token and ClaimableAirdrop smart contracts and deployment scripts.

The review was conducted over three days, from December 17, 2024 to December 19, 2024 by Valentin Quelquejay. A total of 3 person-days were spent.

The system consists of an upgradeable ERC20 token smart contract and an upgradeable ClaimableAirdrop contract, enabling users to claim tokens by providing a valid Merkle proof. Both contracts are deployed behind transparent upgradeable proxies, which are controlled by proxy admins. The foundation's multi-sig wallet owns the proxy admins, as well as the token and airdrop smart contracts. At deployment, a portion of the token supply is minted to a distributor wallet for distribution via the ClaimableAirdrop contract. It is crucial for this wallet to be properly secured. Ideally, it should be a multisig.

During the engagement, the client updated the codebase and addressed all identified issues. No remaining issues were found in the updated version of the codebase. Additionally, the deployments scripts were significantly simplified.

Scope and Objectives

Our initial review focused on the commit hash `4c3e3529218f2cf0af36191d049c103fd0aea98a`. During the engagement, the client updated the commit hash to `a9df9bca1d270fbe87628d1300be84d58d1fed1b`.

The final review focused on two smart contracts and their respective deployment scripts:

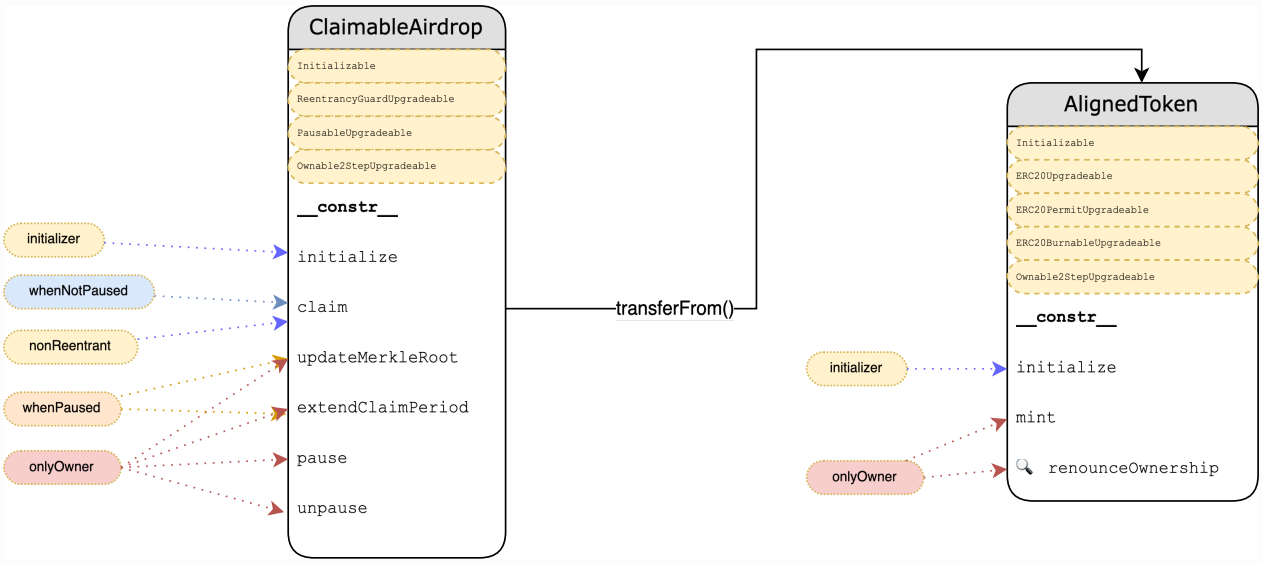
- `claimable_contracts/src/AlignedToken.sol`
- `claimable_contracts/script/DeployAlignedToken.s.sol`
- `claimable_contracts/src/ClaimableAirdrop.sol`
- `claimable_contracts/script/DeployClaimableAirdrop.s.sol`

Together with the Aligned team, we identified the following priorities for our review:

- Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
- Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Security Field Guide, and the ones outlined in the EEA EthTrust Security Levels Specification.
- Ensure the deployment scripts behave as expected.

Audit Artifacts

System Architecture



Findings

Major Incorrect Wallet Address Used in the Deployment Script

Fixed

> The team updated the deployment scripts and addressed this issue in PR 1655.

The function `deployClaimableAirdropProxy` in the script `DeployAll.s.sol` deploys the `ClaimableAirdrop` contract along with a transparent proxy. However, when creating the payload to initialize the `ClaimableAirdrop` implementation contract, it incorrectly sets `_claimSupplier` as `_tokenOwner`, which corresponds to the foundation's multi-sig, instead of `_claim`. As a result, tokens would be distributed from the foundation's wallet instead of the claim supplier's wallet.

claim_contracts_old/script/DeployAll.s.sol

```
131 function deployClaimableAirdropProxy(
132     address _proxyOwner,
133     address _owner,
134     address _tokenOwner,
135     bytes32 _salt,
136     address _deployer,
137     address _token,
138     uint256 _limitTimestampToClaim,
139     bytes32 _claimMerkleRoot
140 ) internal returns (TransparentUpgradeableProxy) {
141     vm.broadcast();
142     ClaimableAirdrop _airdrop = new ClaimableAirdrop();
143
144     bytes memory _airdropDeploymentData = Utils
145         .claimableAirdropProxyDeploymentData(
146             _proxyOwner,
147             address(_airdrop),
148             _owner,
149             _token,
150             _tokenOwner,
151             _limitTimestampToClaim,
152             _claimMerkleRoot
153         );
```

claim_contracts_old/script/Utils.sol

```
154 function claimableAirdropInitData(
155     address _implementation,
156     address _owner,
157     address _tokenContractAddress,
158     address _tokenOwnerAddress,
159     uint256 _limitTimestampToClaim,
160     bytes32 _claimMerkleRoot
```

```
161 ) internal pure returns (bytes memory) {
162     return
163         abi.encodeCall(
164             ClaimableAirdrop(_implementation).initialize,
165             (
166                 _owner,
167                 _tokenContractAddress,
168                 _tokenOwnerAddress,
169                 _limitTimestampToClaim,
170                 _claimMerkleRoot
171             )
172         );
173 }
```

claim_contracts_old/src/ClaimableAirdrop.sol

```
48 function initialize(
49     address _owner,
50     address _tokenProxy,
51     address _claimSupplier,
52     uint256 _limitTimestampToClaim,
53     bytes32 _claimMerkleRoot
54 ) public initializer {
```

Recommendation

Fix the deployment script.

Medium Incorrect Usage of `ReentrancyGuard` instead of `ReentrancyGuardUpgradeable`

Fixed

> The team addressed this issue via PR #1636.

The ClaimableAirdrop contract is using `ReentrancyGuard` instead of `ReentrancyGuardUpgradeable`. Since the contract is upgradeable and inherits from `Initializable` and other upgradeable contracts, one should use the `ReentrancyGuardUpgradeable` version of the reentrancy guard to ensure compatibility with the upgradeable contract pattern. Additionally, it should be initialized in the `initialize` function.

claim_contracts_old/src/ClaimableAirdrop.sol

```
11    contract ClaimableAirdrop is
12        ReentrancyGuard,
13        Initializable,
14        PausableUpgradeable,
15        Ownable2StepUpgradeable
```

Recommendation

Replace `ReentrancyGuard` with its upgradeable version to align with the upgradeable contract pattern and avoid potential future issues. Make sure to initialize it in the `initialize` function by calling the adequate function.

Minor Simplify Deployment Scripts

Fixed

> The team updated and simplified the deployment scripts in PR #1655.

The deployment scripts could be optimized for clarity and consistency. Namely, one should make sure to remove unused functions such as `deployProxyAdmin` in `DeployAll.s.sol` to simplify the code. Additionally, one should make sure to align variable names to match those in the smart contracts, and avoid using different variable names for referring to the same value. This will help to better trace the variables in the scripts and prevent mistakes.

Minor Redundant Modifiers in pause and unpause Functions

Fixed

> The team addressed this issue via PR #1636.

The `pause` and `unpause` functions include both `whenNotPaused`/`whenPaused` modifiers. However, the internal `_pause` and `_unpause` functions already perform the necessary state checks to ensure the contract is not already paused/unpaused. Including these additional modifiers is redundant and increases gas costs without adding value.

claim_contracts_old/src/ClaimableAirdrop.sol

```
139 /// @notice Pause the contract.
140 function pause() public whenNotPaused onlyOwner {
141     _pause();
142 }
143
144 /// @notice Unpause the contract.
145 function unpause() public whenPaused onlyOwner {
146     _unpause();
147 }
```

Recommendation

Remove the `whenNotPaused` and `whenPaused` modifiers from the `pause` and `unpause` functions. The `onlyOwner` modifier is sufficient to restrict access, and the state checks are already handled internally by `_pause` and `_unpause`.

Minor Missing Events in State-Changing Functions

Fixed

> This issue was addressed by the team via PR #1636.

Some state-changing functions do not emit events. Emitting events in state-changing functions is important for logging and monitoring purposes, particularly when tweaking important parameters.

claim_contracts_old/src/ClaimableAirdrop.sol

```
128 function extendClaimPeriod(
129     uint256 newTimestamp
130 ) external whenPaused onlyOwner {
131     require(
132         newTimestamp > limitTimestampToClaim &&
133             newTimestamp > block.timestamp,
134         "Can only extend from current timestamp"
135     );
136     limitTimestampToClaim = newTimestamp;
137 }
```

claim_contracts_old/src/ClaimableAirdrop.sol

```
121 function updateMerkleRoot(bytes32 newRoot) external whenPaused onlyOwner {
122     require(newRoot != 0 && newRoot != claimMerkleRoot, "Invalid root");
123     claimMerkleRoot = newRoot;
124 }
```

Recommendation

Ensure that all state-changing functions emit appropriate events.

None Optional Reentrancy Guard in `ClaimableAirdrop`

Acknowledged

> The team has acknowledged this recommendation and has decided to keep the code as-is for now.

The `ClaimableAirdrop` contract uses a reentrancy guard to protect the `claim` function against reentrancy attacks. However, as long as the function updates the `hasClaimed` mapping before making any external calls, it should be safe from all reentrancy attacks.

claim_contracts/src/ClaimableAirdrop.sol

```
84 function claim(
85     uint256 amount,
86     bytes32[] calldata merkleProof
87 ) external nonReentrant whenNotPaused {
88     require(
89         !hasClaimed[msg.sender],
90         "Account has already claimed the drop"
91     );
```

claim_contracts/src/ClaimableAirdrop.sol

```
105 hasClaimed[msg.sender] = true;
106
107 bool success = IERC20(tokenProxy).transferFrom(
108     tokenDistributor,
109     msg.sender,
110     amount
111 );
```

Recommendation:

The reentrancy guard can be safely removed to save on gas costs. However, keeping it is fine if one prefers an extra layer of protection.

None Redundant Event Emission in `_mint`

Acknowledged

> The team has acknowledged this recommendation and has decided to keep the code as-is for now.

The `_mint` function emits a `TokensMinted` event. The underlying `_mint` function already emits a `Transfer` event with the `from` address set to zero when tokens are minted.

claim_contracts/src/AlignedToken.sol

```
67 function mint(address to, uint256 amount) external onlyOwner {
68     _mint(to, amount);
69     emit TokensMinted(to, amount);
70 }
```

Recommendation

If the `TokensMinted` event is not necessary, it can be removed to reduce gas costs.

None Simplify Initialization of `ClaimableAirdrop`

Fixed

> This issue was addressed via PR #1636.

The `initialize` function of the `ClaimableAirdrop` contract calls `__Ownable2Step_init()` followed by `_transferOwnership(_owner)`. This setup can be simplified by directly using `Ownable_init(_owner)` as the Ownable2Step library uses Ownable under the hood. This will simplify the code, decrease gas usage and improve its readability.

claim_contracts_old/src/ClaimableAirdrop.sol

```
67 __Ownable2Step_init(); // default is msg.sender
68 _transferOwnership(_owner);
```

Recommendation

Replace the combination of `__Ownable2Step_init()` and `_transferOwnership(_owner)` with a single call to `Ownable_init(_owner)`.

File Hashes

- ./script/DeployClaimableAirdrop.s.sol
  - 9c6c5f582f833ea0878b6c9d5117047524b86e0e471e6a11e5a677a13a864b6e
- ./script/DeployAlignedToken.s.sol
  - 7c55a21959fa00107f11e21f2ce602b865307f3aee0d1257a429935ed88b3ecb
- ./src/AlignedToken.sol
  - c4c55c1ab118d044b8b0b10c667de1d9e30c12a6cfcc51770245ae365bc61cd7
- ./src/ClaimableAirdrop.sol
  - 314fbe590cb6c0f3202447b652254dd227b33192f15123a05050ef6689c23b5f

Disclaimer

Creed (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via CD publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that CD are not responsible for the content or operation of such Web sites, and that CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. CD assumes no responsibility for the use of third party software on

the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by CD.