thirdweb A-3

Security Audit

Aug 9th, 2022
Version 1.0.0

Prepared for Matteo

Table of Contents

- Introduction
- Overall Assessment
- Specification
- Source Code
- Issue Descriptions and Recommendations
- Security Levels Reference
- Disclaimer

Introduction

This document includes the results of the security audit for thirdweb's smart contract implementation, revision A-3 of their Drop1155 and Drop721 smart contracts, audited by the Macro security team from June 30, 2022 to June 30, 2022.

The purpose of this audit is to review the source code of smart thirdweb Solidity contracts, and provide information on the design, architecture, and quality of the source code and an emphasis on validating the correctness and security of the software in its entirety.

Disclaimer: While Macro's review is comprehensive and has surfaced some changes that should be made to the source code, this audit should not solely be relied upon for security, as no single audit is guaranteed to catch all possible bugs.

Overall Assessment

The following is an aggregation of issues found by the Macro Audit team:

Severity	Count	Acknowledged	Won't Do	Addressed
High	1	-	-	1
Medium	5	1	2	2
Low	7	2	3	2
Code Quality	5	2	-	3
Informational	2	-	-	2

thirdweb was quick to respond to these issues.

Specification

Our understanding of the specification has based on the following sources:

- Discussions on Slack with the thirdweb team.
- Are audit-hardened documents hosted on Notion.

Source Code

The following source code was reviewed during the audit:

- Repository: contracts
- Current Hash: 0f8d5b0e5f5f5f7a27f2d7b7e8f9e7e5e5d39d

Specifically, we audited the following contracts within this repository:

Contract	SHA256
contracts/prebuilts/drop/Drop1155.sol	5b5d8c3f38b2c9b8d2c3c3f85a2df7f5a2c0c2f7b7b7f8d3f9f7a8d8d8d8d8
contracts/prebuilts/drop/DropERC721.sol	a3f8b7b7d7d7d7a8c7d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8
contracts/lib/CurrencyTransferLib.sol	4f8f8f7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/lib/FeeData.sol	5b5d5f5f7d7d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d
contracts/lib/MerkleProof.sol	7d8f8f7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/lib/TokenERC1155.sol	8d8d8d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/lib/TokenERC721.sol	8d8d8d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/lib/TransparentUpgradeableProxy.sol	9d9d9d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/lib/TWStrings.sol	7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/extension/interface/IDelayedReveal.sol	3f3f3f7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d
contracts/extension/interface/IDelayedReveal.sol	7d8d8d

Not clear how ERC20 accidental transfers would be rejected.

Any function to recover such tokens might affect existing functionality and token balances in the packs.

L-5 Trusted forwarders can be added, not removed

TOPIC	STATUS	IMPACT	LIKELIHOOD
Protocol Longevity	Acknowledged	High	Low

Thirdweb ERC2771Context implementation receives an array of `trustedForwarder` values on initialization. However, there is no way to remove a trusted forwarder.

Consider adding a function to remove a trusted forwarder, or to set the full list of forwarders, to handle cases where forwarders get lost, expired, or compromised.

RESPONSE BY THIRDWEB

> Not fixing (for now).
>
> Today, a thirdweb pre-built smart contract deployed by users on the dashboard or via SDKs use the same set of trusted forwarders deployed by thirdweb, or external providers like Biconomy.
>
> The issue pointed out requires a design change across all of thirdweb's smart contracts; it's fix will be coordinated across the thirdweb products.

L-6 `maxTotalSupply` can be updated to `< totalSupply()`

TOPIC	STATUS	IMPACT	LIKELIHOOD
Input Ranges	Wont Do	Low	Medium

In DropERC20.sol, `setMaxTotalSupply()` allows `maxTotalSupply` to be set to any number. If the value is set to `< totalSupply()` all present and future claims will all fail, and a potentially misleading event will be emitted (`MaxTotalSupplyUpdated()`), potentially causing unnecessary confusion for users. Consider updating `setMaxTotalSupply()` to require a value `>= totalSupply()`.

Note that if the intent is to halt minting immediately, setting `maxTotalSupply` to present `totalSupply()` may suffer from race conditions if additional claims are processed ahead of `setMaxTotalSupply()`. In this case, consider calling `setClaimConditions()` with an empty `_phases` array instead. The latter approach will not revert if additional claims precede its execution.

RESPONSE BY THIRDWEB

> The purpose here was to achieve pausing of claims by setting maxTotalSupply as less than totalSupply().
>
> The absence of checks is intended to transfer responsibility to contract admin.

L-7 Unpublishing may orphan data

TOPIC	STATUS	IMPACT	LIKELIHOOD
Data Model	Acknowledged	Low	Low

The `publishContract()` function adds an entry into `compilerMetadataUriToPublishedMetadataUris` which is not accounted for during `unpublishContract()`.

This may leave orphan `compilerMetadataUri` data, causing it to refer to non-existent `publishMetadataUri` data. Moreover, `getPublishedUriFromCompilerUri()` will return `publishMetadataUris` for contracts that are unpublished.

RESPONSE BY THIRDWEB

> Not fixing (for now).
>
> The issue pointed out requires a design change in the ContractPublisher contract that must be coordinated throughout the thirdweb Release product.

Q-1 Low Test Coverage

TOPIC	STATUS	QUALITY IMPACT
Testing	Acknowledged	High

DropERC20.sol and ContractPublisher.sol have very light testing. Consider adding more to more confidently affirm the behavior of the contracts.

Q-2 Missing Events

TOPIC	STATUS	QUALITY IMPACT
Events	Fixed	Low

- DropERC20.sol's `setContractUri()` does not emit an event.
- Pack.sol's `setPublisherProfileUri()` does not emit an event.

Consider adding an event for these changes.

Q-3 Inconsistent datatype for `platformFeeBps`

TOPIC	STATUS	QUALITY IMPACT
Data Model	Acknowledged	Low

The datatype applied to this value varies throughout the contract between `uint256`, `uint128`, `uint64` and `uint16`. Consider normalizing to the datatype applied to its storage variable definition of `uint128`.

RESPONSE BY THIRDWEB

> Not fixing (yet).
>
> This fix will be coordinated across the relevant thirdweb smart contracts.

Q-4 Unused import

TOPIC	STATUS	QUALITY IMPACT
Extra Code	Fixed	Low

In DropERC20.sol, the import of `ERC20PausableUpgradeable.sol` is not used.

Q-5 Inaccurate `getClaimTimestamp()` return value

TOPIC	STATUS	QUALITY IMPACT
Data Consistency	Fixed	Medium

In DropERC20.sol's `getClaimTimestamp()`, when `lastClaimTimestamp` is zero, `nextValidClaimTimestamp` should arguably be `0` to indicate that no delay is required, since no prior purchase has occurred.

If this were to be implemented, a minor gas-savings opportunity can be implemented along with it, as the `nextValidClaimTimestamp` could be skipped, and line 376 could be simplified to:

```
require(block.timestamp >= nextValidClaimTimestamp, "cannot claim yet.");
```

I-1 `setClaimConditions()` will accept phases that cannot become active

TOPIC	IMPACT
Data Consistency	Informational

`setClaimConditions()` will update `claimCondition` without comparing individual `_phases[i].startTimestamp` values with respect to `block.timestamp`. As a result, it is possible to supply a set of claim conditions which will never execute.

For example, consider calling `setClaimCondition` with 4 entries in `_phases`, with `startTimestamp` values of: `50`, `75`, `100`, `125`. If executed when `block.timestamp = 100`, the first two phases will never occur.

This may impact an owner's intended phases, or mask accidental errors in phase formulation.

Disclaimer

Macro makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and Macro specifically disclaims all implied warranties of merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

Macro will not be liable for any lost profits, business, contracts, revenue, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand by any other party. In no event will Macro be liable for consequential, incidental, special, indirect, or exemplary damages arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if Macro has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the Emergent team and only the source code Macro notes as being within the scope of Macro's review within this report. This report does not include an audit of the deployment scripts used to deploy the Solidity contracts in the repository corresponding to this audit. 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. In this report you may through hypertext or other computer links, gain access to websites operated by persons other than Macro. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such websites' owners. You agree that Macro is not responsible for the content or operation of such websites, and that Macro shall have no liability to your or any other person or entity for the use of third party websites. Macro assumes no responsibility for the use of third party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.