Protocol Audit Report

Version 1.0

Tura11

February 21, 2026

Protocol Audit Report

Tura11

February 21 february 2026

Prepared by: Tura11 Lead Auditor: Tura11

Table of Contents

Protocol Summary

Santa's List is the main contract that stores the list of naughty and nice people. It doubles as an NFT contract that people can collect if they are `NICE` or `EXTRA_NICE`. In order for someone to be considered `NICE` or `EXTRA_NICE` they *must* be first "checked twice" by Santa.

Once they are checked twice, `NICE` users can collect their NFT, and `EXTRA_NICE` users can collect their NFT **and** they are given `SantaToken`s. The `SantaToken` is an ERC20 that can be used to buy the NFT for their `NAUGHTY` or `UNKNOWN` friends.

List Checking

In this contract **Only Santa** to take the following actions: - `checkList`: A function that changes an `address` to a new `Status` of `NICE`, `EXTRA_NICE`, `NAUGHTY`, or `UNKNOWN` on the *original* `s_theListCheckedOnce` list. - `checkTwice`: A function that changes an `address` to a new `Status` of `NICE`, `EXTRA_NICE`, `NAUGHTY`, or `UNKNOWN` on the *new* `s_theListCheckedTwice` list **only** if someone has already been marked on the `s_theListCheckedOnce`.

SantaToken

This codebase is based off solmate a Modern, opinionated, and gas optimized building blocks for smart contract development. The ERC20 is a typical ERC20 with the following changes: - Only `SantasList` can mint tokens - Only `SantasList` can burn tokens (well, technically anyone can, but only `SantasList` can call the burn function)

TokenUri.sol

A minimal contract that exclusively has the tokenURI. It's a separate contract inherited by `SantasList` for readability purposes.

Disclaimer

The Tura11 team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

- Commit Hash: 91c8f8c94a9ff2db91f0ab2b2742cf1739dd6374
- In Scope:

Scope

```
./src/
#-- SantaToken.sol
#-- SantasList.sol
#-- TokenUri.sol
```

Roles

- `Santa` - Deployer of the protocol, should only be able to do 2 things:
  - `checkList` - Check the list once
  - `checkTwice` - Check the list twice
  - Additionally, it's OK if Santa mints themselves tokens.
- `User` - Can buyPresents and mint NFTs depending on their status of NICE, NAUGHTY, EXTRA-NICE or UNKNOWN

Executive Summary

Ive learnt a lot from this audit, process was pretty easliy and fast. ## Issues found’

Severity	Number of issues found
Critical	1
High	3
Medium	0
Low	0
Info	0
Gas	0
Total	4

Findings

Critical

[C-1] business logic in BuyPresent function, leads to drain user funds

High

[H-1]Access control

[H-2] All addresses are Nice by deafault

[H-3]Nice or Extra nice are able to call collectPresent function multiple times