Boss Bridge

This project presents a simple bridge mechanism to move our ERC20 token from L1 to an L2 we're building. The L2 part of the bridge is still under construction, so we don't include it here.

In a nutshell, the bridge allows users to deposit tokens, which are held into a secure vault on L1. Successful deposits trigger an event that our off-chain mechanism picks up, parses it and mints the corresponding tokens on L2.

To ensure user safety, this first version of the bridge has a few security mechanisms in place:

- The owner of the bridge can pause operations in emergency situations.
- Because deposits are permissionless, there's an strict limit of tokens that can be deposited.
- Withdrawals must be approved by a bridge operator.

We plan on launching `L1BossBridge` on both Ethereum Mainnet and ZKSync.

Token Compatibility

For the moment, assume *only* the `L1Token.sol` or copies of it will be used as tokens for the bridge. This means all other ERC20s and their weirdness is considered out-of-scope.

On withdrawals

The bridge operator is in charge of signing withdrawal requests submitted by users. These will be submitted on the L2 component of the bridge, not included here. Our service will validate the payloads submitted by users, checking that the account submitting the withdrawal has first originated a successful deposit in the L1 part of the bridge.

Getting Started

Requirements

- git
  - You'll know you did it right if you can run `git --version` and you see a response like `git version x.x.x`
- foundry

- You'll know you did it right if you can run `forge --version` and you see a response like `forge 0.2.0 (816e00b 2023-03-16T00:05:26.396218Z)`

Quickstart

```
git clone https://github.com/Cyfrin/7-boss-bridge-audit
cd 7-boss-bridge-audit
make
```

or

```
git clone https://github.com/Cyfrin/7-boss-bridge-audit
cd 7-boss-bridge-audit
forge install
forge build
```

Usage

Testing

```
forge test
```

Test Coverage

```
forge coverage
```

and for coverage based testing:

```
forge coverage --report debug
```

Static Analysis

Slither

```
make slither
```

Aderyn

```
make aderyn
```

Audit Scope Details

- Commit Hash: 07af21653ab3e8a8362bf5f63eb058047f562375
- In scope

```
./src/
#-- L1BossBridge.sol
#-- L1Token.sol
#-- L1Vault.sol
#-- TokenFactory.sol
```

- Solc Version: 0.8.20
- Chain(s) to deploy contracts to:
    - Ethereum Mainnet:
        - L1BossBridge.sol
        - L1Token.sol
        - L1Vault.sol
        - TokenFactory.sol
    - ZKSync Era:
        - TokenFactory.sol
    - Tokens:
        - L1Token.sol (And copies, with different names & initial supplies)

Actors/Roles

- Bridge Owner: A centralized bridge owner who can:
    - pause/unpause the bridge in the event of an emergency
    - set `Signers` (see below)
- Signer: Users who can "send" a token from L2 -> L1.
- Vault: The contract owned by the bridge that holds the tokens.
- Users: Users mainly only call `depositTokensToL2`, when they want to send tokens from L1 -> L2.

Known Issues

- We are aware the bridge is centralized and owned by a single user, aka it is centralized.
- We are missing some zero address checks/input validation intentionally to save gas.
- We have magic numbers defined as literals that should be constants.
- Assume the `deployToken` will always correctly have an L1Token.sol copy, and not some weird erc20