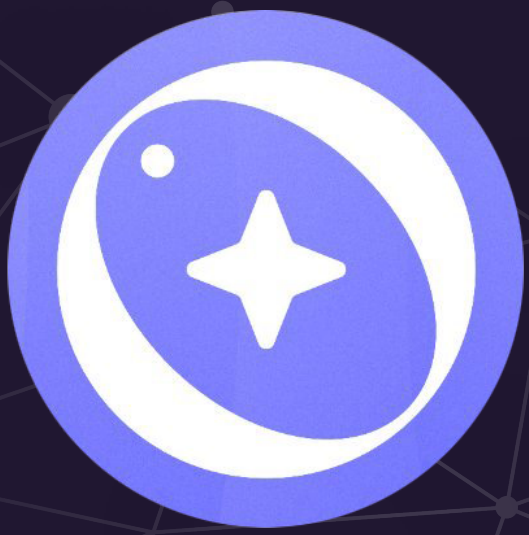
SB SECURITY

Tierra

Security Review



April 28, 2025

Contents

1. About SBSecurity 3
2. Disclaimer 3
3. Risk classification 3
   - 3.1. Impact 3
   - 3.2. Likelihood 3
   - 3.3. Action required for severity levels 3
4. Executive Summary 4
5. Findings 5
   - 5.1. Medium severity 5
     - 5.1.1. TierraDAO::finalizeFundraising can be DoS 5
   - 5.2. Low severity 6
     - 5.2.1. Locker deployment is missing access control 6
     - 5.2.2. Use call instead of transfer 6
     - 5.2.3. TRR tokens can be reused by multiple contributors 6
     - 5.2.4. Missing setter for the liquidityLockerFactory 7
     - 5.2.5. TierraDAOLaunchedToken owner is limited to 1% 7
   - 5.3. Governance severity 8
     - 5.3.1. Locker deployment is missing access control 8

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter @Slavcheww.

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1. Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.

4. Executive Summary

Tierra Dao contracts have been audited through the Hyacinth platform.

Overview

Project	Tierra
Repository	Private
Commit Hash	42d2857da1cda9a5d2af13ec9608e9c643e25902
Resolution	5371cc5672f275388c4039eeada654d6423f4f6d
Timeline	April 24, 2025 - April 25, 2025

Scope

TierraDAOLaunchedToken.sol
TierraDAOFactory.sol
TierraDAO.sol
LpLocker.sol
LockerFactory.sol

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	1
Low/Info Risk	5
Governance Risk	1

5. Findings

5.1. Medium severity

5.1.1. `TierraDAO::finalizeFundraising` can be DoS

Severity: Medium Risk

Description: There's no minimum contribution amount, and adversaries can utilize that to add many records (addresses) in the `contributors` array with as little as 1 wei. After that, if the fundraising goal has been reached, `finalizeFundraising` will revert with OOG, because of the for-loop, which is responsible for minting DAO tokens to the contributors:

```
function finalizeFundraising(int24 initialTick, int24 upperTick, bytes32 salt) external onlyFactory {
    ...MORE CODE
    for (uint256 i = 0; i < contributors.length; i++) {
        address contributor = contributors[i];
        uint256 contribution = contributions[contributor];
        uint256 tokensToMint = (contribution * SUPPLY_TO_FUNDRAISERS) / totalRaised;

        token.mint(contributor, tokensToMint);
  }
  ...MORE CODE
}
```

The fact that the function is already gas-intensive as it deploys 3 contracts - the DAO token, the LPLocker, and the UniV3 will make the attack more feasible.

If they manage to fill the contribution so that it exceeds the block gas limit as well as fill the fundraising goal, the presale is blocked as well as the refund, which has the following require:

```
function refund() external nonReentrant {
    require(!goalReached, "Fundraising goal was reached");
    ...MORE CODE
}
```

Note that this can happen even unintentional, when the `fundraisingGoal` is relatively big and there's a lot of interest in this particular DAO.

Recommendation: Consider extracting the DAO token claiming logic into a separate function, so it behaves in a pull-over-push manner, but make sure to add the check in the `_update` function of the `TierraDAOLaunchedToken`. Also, consider adding a minimal contribution amount.

Resolution: Fixed

5.2. Low severity

5.2.1. Locker deployment is missing access control

Severity: Low Risk

Description: `LockerFactory::deploy` is missing access-control and anyone can deploy new LP lockers. Although no issues were observed if someone does that, it can potentially show false information on the UI of the `TierraDAO` if there's a listener for the `deployed` event.

Recommendation: Add access control to the function so only valid DAOs can call deploy. You can use `isFund` mapping in the `TierraDAOFactory`.

Resolution: Acknowledged

5.2.2. Use call instead of transfer

Severity: Low Risk

Description: Using `transfer` instead of `call` in `TierraDAO::refund` and `TierraDAO::contribute` to return BERA to contributors can cause issues if the recipients are smart contracts or smart wallets such as Gnosis Safe. The reason is that `transfer` forwards only 2300 gas, which is insufficient to execute the receive logic of the GnosisSafe wallet, for example.

Recommendation: Use `call` over `transfer`, but make sure to follow the CEI pattern and execute the call after the entire contract state is updated.

Resolution: Fixed

5.2.3. TRR tokens can be reused by multiple contributors

Severity: Low Risk

Description: Although not entirely related to the TierraDAO, currently `TRR` tier whitelist can be gamed from contributors by transferring the tokens between multiple addresses, so they can bypass the `TRR_AMOUNT_TO_BE_WHITELISTED`.

```
ERC20(TRR).balanceOf(msg.sender) >= TRR_AMOUNT_TO_BE_WHITELISTED
```

Recommendation: There's nothing that can be done in `TierraDAO` in order to avoid it.

Resolution: Acknowledged

5.2.4. Missing setter for the `liquidityLockerFactory`

Severity: Low Risk

Description: `TierraDAO` is missing a setter function for the `liquidityLockerFactory` in case new one gets deployed while the fundraising takes place.

Recommendation: Consider adding a setter function for the `LPLocker` factory in `TierraDAO`.

Resolution: Acknowledged. We would not be likely to deploy a new factory while fund raising is happening.

5.2.5. TierraDAOLaunchedToken owner is limited to 1%

Severity: Low Risk

Description: `TierraDAOLaunchedToken` has whale protection in the first hour of the pool. Comments say that the owner of the token, which is `TierraDAO`, should not be limited to this, but in fact it is.

```
function _update(address from, address to, uint256 amount) internal virtual override {
    super._update(from, to, amount);

    // Prevents any wallet from buying too many tokens right at LP launch
    // Permanently disabled afterwards
    if (lubricating) {
        if (block.timestamp - lubricationStartTime > lubricationLength) {
            lubricating = false;

            // Allow deployer (owner) to send/receive any amount and the liquidityPool to receive any
amount. // AUDIT - owner cannot receive any amount
            // This allows for loading of the LP, and for people to sell tokens into the LP whilst
lubrication in progress.
        } else if (from != owner() && to != liquidityPool) {
            // Require that a receiving wallet will not hold more than 1% of supply after a transfer whilst
lubrication is in effect
            require(balanceOf(to) <= totalSupply() / 100, "Warmup! Please buy less tokens");
        }
    }
}
```

Recommendation: Even that the owner will not have tokens in its balance, if that's your goal, you can easily do it. Just add `&& to != owner()`.

Resolution: Acknowledged

5.3. Governance severity

5.3.1. Locker deployment is missing access control

Severity: Governance Risk

Description: If the `owner` or `protocolAdmin` have their key stolen, an attacker can:

- Steal all the funds from the TierraDao contracts with `execute` functions.

- `emergencyEscape` also can be used to sweep all the BERA.

Recommendation: Add mappings with the allowed function signatures and addresses that can be called.

Resolution: Acknowledged. Protocol admin will sit behind a HW wallet and is only relevant during raising after that owner and factory will sit behind a 2/3 multisig.