Superseed: Token Contracts

Security Review

Cantina Managed review by:

0xWeiss, Security Researcher
Kankodu, Security Researcher

April 14, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Superseed is a network that transforms Ethereum scaling into self-repaying loans.

From Mar 25th to Mar 26th the Cantina team conducted a review of superseed-token-contracts on commit hash 3f7f8947. The team identified a total of **6** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	2	2	0
Gas Optimizations	1	1	0
Informational	3	3	0
Total	**6**	**6**	**0**

3 Findings

3.1 Low Risk

3.1.1 Missing input validation in constructors

Severity: Low Risk

Context: TokenClaim.sol#L43, SuperseedToken.sol#L31

Description: Both contracts in scope (TokenClaim and SuperseedToken) lack input validation on their constructors.

Recommendation: Validate the following parameters:

- `TokenClaim`:

```
    constructor(address _initialOwner, address _token, address _treasury) Ownable(_initialOwner) {
+       require(_token != address(0), "invalid address");
+       require(_treasury != address(0), "invalid address");

        token = IERC20(_token);
        treasury = _treasury;
    }
```

- `SuperseedToken`:

```
    constructor(
        address superAdmin,
        address minter,
        address treasury
    )
        ERC20("Superseed", "SUPR")
        ERC20Permit("Superseed")
    {
+       require(treasury != address(0), "invalid address");
+       require(superAdmin != address(0), "invalid address");
+       require(minter != address(0), "invalid address");
        _mint(treasury, 10_000_000_000e18);

        _grantRole(DEFAULT_ADMIN_ROLE, superAdmin);
        _grantRole(MINTER_ROLE, minter);
    }
```

Superseed: Fixed in commits 9cc97c94 and 69e15ade.

Cantina Managed: Fix verified.

3.1.2 Missing event when updating the merkle root

Severity: Low Risk

Context: TokenClaim.sol#L94

Description: The merkle root function makes a key state change but lacks to emit an event:

```
function setMerkleRoot(bytes32 _merkleRoot) external onlyOwner {
    if (_merkleRoot == bytes32(0)) revert InvalidInput("_merkleRoot");

    merkleRoot = _merkleRoot;
}
```

Recommendation: Emit an event with the old and the new merkle roots as arguments:

```
  function setMerkleRoot(bytes32 _merkleRoot) external onlyOwner {
      if (_merkleRoot == bytes32(0)) revert InvalidInput("_merkleRoot");

+     emit merkleRootUpdated(merkleRoot,_merkleRoot);

      merkleRoot = _merkleRoot;
  }
```

Superseed: Fixed in commit 69e15ade.

Cantina Managed: Fix verified.

3.2 Gas Optimization

3.2.1 Improve Custom Error Definitions for Clarity and Efficiency

Severity: Gas Optimization

Context: TokenClaim.sol#L35, TokenClaim.sol#L82, TokenClaim.sol#L92

Description: The `InvalidInput` error currently includes a string parameter, which increases bytecode size and provides no significant advantage over using a standard `revert` with a string message.

Recommendation: Instead of using a string parameter, define multiple custom errors with structured and informative parameters. This improves clarity while reducing bytecode size. For example:.

```
error MerkleRootCannotBeEmpty();
error ZeroBalanceForProvidedToken(address token);
error InputAmountCannotBeZero();
```

By using structured parameters, errors become more meaningful and cost-efficient while maintaining clarity.

Superseed: Fixed in commit 69e15ade.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 ERC20 import is redundant

Severity: Informational

Context: SuperseedToken.sol#L20

Description: The `SuperseedToken` contract imports multiple ERC20 extensions: `ERC20`, `ERC20Burnable`, `AccessControl`, `ERC20Permit`, `ERC20Votes` which under the hood already use the original `ERC20` contract, making it redundant.

Recommendation: Remove the `ERC20` import:

```
- contract SuperseedToken is ERC20, ERC20Burnable, AccessControl, ERC20Permit, ERC20Votes {
+ contract SuperseedToken is ERC20Burnable, AccessControl, ERC20Permit, ERC20Votes {
```

Superseed: Fixed in commit 9cc97c94.

Cantina Managed: Fix verified.

3.3.2 Avoid Unnecessary Use of `SafeERC20` for `SuperseedToken`

Severity: Informational

Context: TokenClaim.sol#L14, TokenClaim.sol#L75

Description: Since `SuperseedToken` strictly follows the ERC20 standard and reverts on failed transfers, the safety checks provided by the SafeERC20 library are unnecessary.

Recommendation: Use direct ERC20 `transferFrom` call to avoid unnecessary overhead.

Superseed: Fixed in commit 69e15ade.

Cantina Managed: Fix verified.

3.3.3 Use Named Constants for Improved Readability

Severity: Informational

Context: SuperseedToken.sol#L41

Description: Named constants enhance code clarity and maintainability by replacing hardcoded values with meaningful identifiers.

Recommendation: Define `TEN_BILLION_TOKENS` as a constant:.

```
uint256 internal constant TEN_BILLION_TOKENS = 10_000_000_000e18;
```

Use this constant instead of directly writing the value to improve readability and prevent errors.

Superseed: Fixed in commit 9cc97c94.

Cantina Managed: Fix verified.