SALUS SECURITY

JUL 2025

CODE SECURITY ASSESSMENT

CYCLE

Overview

Project Summary

- Name: CycleNetwork Token
- Version: v1
- Platform: EVM-compatible chains
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	CycleNetwork Token
Version	v2
Type	Solidity
Dates	Jul 11 2025
Logs	Jul 11 2025; Jul 11 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	2
Total	3

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk with initial token distribution	Low	Centralization	Mitigated
2	Magic numbers are used	Informational	Code Quality	Acknowledge
3	Use of floating pragma	Informational	Configuration	Acknowledge

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk with initial token distribution	
Severity: Low	Category: Centralization
Target: - contracts/CycleNetworkToken.sol	

Description

contracts/CycleNetworkToken.sol:L9 - L17

```
constructor(
    string memory _name,
    string memory _symbol,
    address _lzEndpoint,
    address _delegate,
    address _receiver
) OFT(_name, _symbol, _lzEndpoint, _delegate) Ownable(msg.sender) ERC20Permit(_name) {
    _mint(_receiver, 1_000_000_000 * 1e18);
}
```

When the contract is deployed, `` `$CYC` `` is sent to one account. This account then has full control over the token distribution. If it is an EOA account, any compromise of its private key could drastically affect the project – for example, attackers could manipulate the price of `` `$CYC` `` on the DEX if they gain access to the private key.

Recommendation

It is recommended to transfer tokens to a multi-sig account and promote transparency by providing a breakdown of the intended initial token distribution in a public location.

Status

The team transfers the owner address to a multi-sig account after deployment.

2.3 Informational Findings

2. Magic numbers are used	
Severity: Informational	Category: Code Quality
Target: - contracts/CycleNetworkToken.sol	

Description

There are a few occurrences of literal values being used. To improve the code's readability and facilitate refactoring, consider defining a constant for every magic number, giving it a clear and self-explanatory name.

contracts/CycleNetworkToken.sol:L16

```
_mint(receiver, 1_000_000_000 * 1e18);
```

Recommendation

Consider defining a constant variable `initialSupply`for the magic number `1_000_000_000 * 1e18`.

Status

This issue has been acknowledged by the team.

3. Use of floating pragma	
Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.22;
```

All contracts use a floating compiler version `^0.8.22`.

Using a floating pragma `^0.8.22`statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file in commit 6982e88:

File	SHA-1 hash
contracts/CycleNetworkOFT.sol	32745493f7d4aa0dcbb7afee32baae1510633454
contracts/CycleNetworkToken.sol	22ee80a6b5d0744c12e24c4f2a8e341c50c4faef