SALUS SECURITY

JUN 2025

CODE SECURITY ASSESSMENT

TOKENUP

Overview

Project Summary

- Name: Tokenup - card contract
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/web3-tokenup/card_contract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Tokenup - card contract
Version	v2
Type	Solidity
Dates	Jun 23 2025
Logs	Jun 18 2025; Jun 23 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	3
Total	4

Contact

E-mail: support@salusec.io

SALUS

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.

SALUS

Content

SALUS

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.

SALUS

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Low	Centralization	Acknowledged
2	Missing depositor address in deposit event	Informational	Logging	Acknowledged
3	Missing two-step transfer ownership pattern	Informational	Business logic	Acknowledged
4	Unused tokens array	Informational	Redundancy	Acknowledged

SALUS

2.2 Notable Findings

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

1. Centralization risk

Severity: Low	Category: Centralization
Target: - contracts/card_service.sol	

Description

The `card_service.sol` contract has privileged accounts. These privileged accounts can withdraw all tokens from the contracts using the `withdraw()` function.

If privileged accounts' private key or admin's is compromised, an attacker can steal all the tokens in the contract.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

SALUS

2.3 Informational Findings

2. Missing depositor address in deposit event

Severity: Informational	Category: Logging
Target: - contracts/card_service.sol	

Description

The `Deposit` event is declared as `event Deposit(uint256 amount, address tokenAddress);` but does not include the `msg.sender` field. Without indexing the depositor's address, on-chain event logs cannot easily attribute deposits to specific accounts.

Recommendation

Modify the event signature to include and index the depositor address, for example:

```
event Deposit(address indexed sender, uint256 amount, address indexed tokenAddress);
```

Status

This issue has been acknowledged by the team.

SALUS

3. Missing two-step transfer ownership pattern

Severity: Informational	Category: Business logic
Target: - contracts/card_service.sol	

Description

The `card_service` contract inherits from the Ownable contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

This issue has been acknowledged by the team.

SALUS

4. Unused tokens array

Severity: Informational	Category: Redundancy
Target: - contracts/card_service.sol	

Description

The tokens state variable `address[] tokens` is never read from or written to anywhere in the contract's logic. As a result, it burns storage slots without providing any functional benefit.

contracts/card_service.sol:L16

```
address[] tokens;
```

Recommendation

Remove the tokens array declaration.

Status

This issue has been acknowledged by the team.

SALUS

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 0a0c637:

File	SHA-1 hash
contracts/card_service.sol	1d00a76eb927604d823af77fdc202381c4154094

SALUS