SALUS SECURITY

JAN 2024

CODE SECURITY ASSESSMENT

T3 FINANCE

Overview

Project Summary

- Name: T3 Finance - TMX token
- Platform: Optimism
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	T3 Finance - TMX token
Version	v2
Type	Solidity
Dates	Jan 15 2024
Logs	Jan 05 2024; Jan 15 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	0
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	Medium	Centralization	Acknowledged
2	Missing events	Informational	Logging	Acknowledged
3	Missing zero address checks	Informational	Data Validation	Acknowledged

2.2 Notable Findings

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

1. Centralization risk	
Severity: Medium	Category: Centralization
Target: - contracts/tokens/BaseToken.sol	

Description

1. The permissions granted to the gov user are excessively broad.

The gov address in the contract can set all other roles, including Admin, Handler, and Minter. Meanwhile, it can also modify important variables and withdraw tokens in the contract.

However, when the contract is deployed, gov is set as an EOA account. If the gov's private key is compromised by an attacker, the attacker has the authority to make arbitrary settings on the contract, and mint/burn any user's tokens with the Minter role.

2. The Handler can transfer anyone's tokens without approval.

The Handler role set by the gov belongs to the token transfer whitelist user.

In the transferFrom() function, any user's token can be transferred by the Handler without approval from the token owner. When private transfer mode is on, only the Handler can transfer tokens. If the Handler's private key is compromised, the attacker can transfer anyone's tokens at will.

contracts/tokens/BaseToken.sol:L144-L147

```
function transferFrom(address _sender, address _recipient, uint256 _amount) external
override returns (bool) {
        if (isHandler[msg.sender]) {
            _transfer(_sender, _recipient, _amount);
            return true;
        }
        ...
}
```

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.

2.3 Informational Findings

2. Missing events

Severity: Informational	Category: Logging
Target: - contracts/tokens/BaseToken.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

However, all setting functions that modify variables in the contract do not set events. Especially the following functions, the modification of these variables is related to user asset management.

contracts/tokens/BaseToken.sol:L61-63, L78-L80, L82-84

```
function setYieldTrackers(address[] memory _yieldTrackers) external onlyGov {
        yieldTrackers = _yieldTrackers;
}

function setInPrivateTransferMode(bool _inPrivateTransferMode) external override onlyGov
{
       inPrivateTransferMode = _inPrivateTransferMode;
}

function setHandler(address _handler, bool _isActive) external onlyGov {
        isHandler[_handler] = _isActive;
}
```

Recommendation

Consider designing appropriate events and incorporating them into the important setting functions.

Status

This issue has been acknowledged by the team.

3. Missing zero address checks

Severity: Informational	Category: Data Validation
Target: - contracts/tokens/BaseToken.sol	

Description

It is considered a security best practice to verify addresses against the zero address in the constructor or setting. However, this precautionary step is absent for the variables highlighted below. If the gov address is set to 0, it is no longer possible to use onlyGov functions.

contracts/tokens/BaseToken.sol:L52-L54

```
function setGov(address _gov) external onlyGov {
        gov = _gov;
}
```

Recommendation

Consider adding zero-address checks.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files from address 0x916B0bB4A98a3d72FCB1c2E67eBaCcf7ac47D7f5:

File	SHA-1 hash
GMX.sol	c6098ec0ec40857f1542869b22ad5b1e094640d7
BaseToken.sol	2ab07293ea5f7fd49f2ad31026fcb097db56c283
MintableBaseToken.sol	12e55334d799c6a9b24e7f99b7208bd409097b97