SALUS SECURITY

DEC 2022

CODE SECURITY ASSESSMENT

TOPGOAL

Overview

Project Summary

- Name: TopGoal
- Version: v1.0.0
- Platform: BSC
- Language: Solidity
- Repository: https://github.com/topgoalnft/goal-contract/tree/4a2d368f222d6548c73635c1e5cdfe0c2d63ccf8
- Audit Scope: See Appendix A

Project Dashboard

Application Summary

Name	TopGoal
Version	v2
Type	Solidity
Dates	Dec 23 2022
Logs	Dec 23 2022

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

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	Missing whenNotPaused modifier	Low	Access Control	Resolved
2	Constructor visibility not needed	Informational	Redundancy	Resolved
3	Code with no effect	Informational	Redundancy	Resolved
4	SafeMath library not needed	Informational	Redundancy	Acknowledged

2.2 Notable Findings

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

1. Missing whenNotPaused modifier

Severity: Low	Category: Access Control
Target: - contracts/TopGoalToken.sol	

Description

contracts/TopGoalToken.sol:L111, L124

```
function delegate(address delegatee) external {...}

function delegateBySig(
    address delegatee,
    uint nonce,
    uint expiry,
    uint8 v,
    bytes32 r,
    bytes32 s
)
    external
{...}
```

In **TopGoalToken**, the **_beforeTokenTransfer**() function contains the **whenNotPaused** modifier, while **delegate**() and **delegateBySig**() do not.
All critical external functions should contain the **whenNotPaused** modifier to stop transactions while paused.

Recommendation

Consider adding the **whenNotPaused** modifier to **delegate**() and **delegateBySig**()

```
function delegate(address delegatee) whenNotPaused external {...}

function delegateBySig(
    address delegatee,
    uint nonce,
    uint expiry,
    uint8 v,
    bytes32 r,
    bytes32 s
)
    whenNotPaused
    external
{...}
```

Status This issue has been fixed in the commit: a113c56.

2.3 Informational Findings

2. Constructor visibility not needed	
Severity: Informational	Category: Redundancy
Target: - contracts/BEP20.sol - contracts/TopGoalToken.sol	

Description

Constructors no longer require visibility (public or external) since Solidity 0.7.0. Therefore, the public modifier in the following constructors can be removed.

contracts/BEP20.sol:L59

```
constructor(string memory name_, string memory symbol_) public {...}
```

contracts/TopGoalToken.sol:L55

```
constructor()
    BEP20("TopGoal Token", "Goal") public
{...}
```

Status This issue has been fixed in the commit: a113c56.

3. Code with no effect	
Severity: Informational	Category: Redundancy
Target: - contracts/TopGoalToken.sol	

Description

contracts/TopGoalToken.sol:L44-L50

```
constructor()
    BEP20("TopGoal Token", "Goal") public
{
    uint256 amount = 1000000000 * 10 ** 18;
    _mint(msg.sender, amount);
    _moveDelegates(address(0), _delegates[msg.sender], amount);
}
```

The **_moveDelegates**(..) line has no effect; therefore, it can be removed.

Status This issue has been fixed in the commit: a113c56.

4. SafeMath library not needed	
Severity: Informational	Category: Redundancy
Target: - contracts/BEP20.sol	

Description

contracts/BEP20.sol:L8

```
import '@openzeppelin/contracts/utils/math/SafeMath.sol';
```

SafeMath is used to check underflow and overflow for arithmetic operations. However, since Solidity version 0.8.0, arithmetic operations revert on underflow and overflow by default. Since **BEP20** uses a Solidity version no less than 0.8.0, it is unnecessary to use the **SafeMath** library.

Status This issue has been acknowledged by the team.

Appendix

Appendix A - Files in Scope

This audit covered the following files:

File	SHA-1 hash
contracts/BEP20.sol	9f172fe2fc18805a4275eb224e4523ad8c641b7e
contracts/IBEP20.sol	d3d5fb23a1d6246aa1e284a1e38d5210a8e08217
contracts/TopGoalToken.sol	8bf1fb7d79fab9a4b5fd5a3c1838476f5969a90b