SALUS SECURITY

AUG 2023

CODE SECURITY ASSESSMENT

LIFEFORM

Overview

Project Summary

- Name: Lifeform - LFT Token
- Platform: BNB Smart Chain
- Language: Solidity
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	Lifeform - LFT Token
Version	v1
Type	Solidity
Dates	Aug 16 2023
Logs	Aug 16 2023

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	Centralization risk	Low	Centralization	Pending
2	Missing zero-address checks	Informational	Data Validation	Pending
3	Redundant code	Informational	Redundancy	Pending
4	Gas optimization suggestions	Informational	Gas optimization	Pending

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/Claimable.sol	

Description

In the Claimable, there is a privileged Owner role. This role has the ability to:

- Set the claimer's address
- Pause the contract
- Withdraw the entire amount that the claimer should receive.

If an attacker were to gain access to the private keys associated with this role, they could cause significant damage to the project.

Recommendation

Consider transferring the privileged roles to multi-sig accounts.

2.3 Informational Findings

2. Missing zero-address checks

Severity: Informational	Category: Data Validation

Target:
- contracts/Claimable.sol
- contracts/LifeformToken.sol

Description

It is considered a security best practice to verify addresses against the zero address during initialization or setting. However, this safeguard is currently missing for address variables.

contracts/LifeformToken.sol:L44,L50

```
function addBlackAccount(address blackAccount) external onlyOwner {
    ...
}

function delBlackAccount(address blackAccount) external onlyOwner {
    ...
}
```

contracts/Claimable.sol:L26,L33

```
constructor( address owner, IERC20 erc20Token, uint256 tgeTimestamp){
    ...
}


function setClaimer(address claimer) public onlyOwner{
    ...
}
```

Recommendation

Consider adding zero address checks for address variables.

3. Redundant code

Severity: Informational	Category: Redundancy

Target:
- contracts/IftReserveCap2.sol
- contracts/LifeformToken.sol

Description

1.The _mint() function already has a similar event emission within it, hence there is no need to trigger a separate Mint() event in the mint() function.

contracts/LifeformToken.sol:L31

```
function mint(address to, uint256 amount) public onlyOwner returns(bool) {
    ...
    _mint(to, amount);
    emit Mint(address(0), to, amount);
    ...
}
```

2.In the vestedAmount(), the condition statement `if(timestamp < _tgeTimestamp + steps * (90 days))` and the return statement `return CAP_RESERVE_FUND;` can be omitted, as the result obtained from `return supplyPerQuarter * steps;` would be the same value.

contracts/IftReserveCap2.sol:L28-L32

```
function vestedAmount(uint256 timestamp) public view override returns (uint256) {
    ...
     if(timestamp < _tgeTimestamp + steps * (90 days)){
        return supplyPerQuarter * steps;
    }

    return CAP_RESERVE_FUND;
}
```

3.The urgencyWithdrawGasToken() function allows the owner to transfer the contract's ETH balance to a specified target address. However, both the Claimable contract and the IftReserveCap contract do not have any payable functions or receive function to accept ETH. As a result, the urgencyWithdrawGasToken() seems to be unnecessary.

contracts/Claimable.sol:L47-L49

```
function urgencyWithdrawGasToken(address target) public onlyOwner {
    payable(target).transfer(address(this).balance);
}
```

Recommendation

Consider removing the redundant code.

4. Gas optimization suggestions

Severity: Informational	Category: Gas Optimization
Target: - contracts/lftReserveCap4.sol - contracts/lftReserveCap2.sol	

Description

1. _startTimestamp and _endTimestamp are only set in the constructor. Thus, they can be marked as immutable to save gas.

contracts/lftReserveCap4.sol:L11-L12

```
uint256 public _startTimestamp;
uint256 public _endTimestamp;
```

2. The variable _tgeTimestamp is repeatedly accessed from storage, which is gas inefficient. To optimize, retrieve its value from storage once, store it in a local variable, and use this local variable in subsequent instances. This approach will save gas.

contracts/lftReserveCap2.sol:L16-L34

```
function vestedAmount(uint256 timestamp) public view override returns (uint256) {

    require(timestamp >= _tgeTimestamp, "LFT: timestamp before TGE timestamp");

    uint256 steps = 1 + (timestamp - _tgeTimestamp) / (90 days);

    ...

    if(timestamp < _tgeTimestamp + steps * (90 days)){
        return supplyPerQuarter * steps;
    }
    ...
}
```

Recommendation

Consider implementing the gas-saving tips mentioned above..

Appendix

Appendix 1 - Files in Scope

This audit covered the following files provided by the client:

File	SHA-1 hash
Claimable.sol	4cd29976ec291f026769ab9b869094b372b73df2
IftReserveCap2.sol	0814b498a1e1b16bcfb747cdcdc821017a96ae0e
IftReserveCap4.sol	2be5ad81ccfe8105ec5a4850522c2a6c46fc5748
LifeformToken.sol	cfb2c29c86db3d4aebf3f3622eaf25b7b607854f