SALUS SECURITY

OCT 2023

CODE SECURITY ASSESSMENT

JUNGLE(JNGL) TOKEN

Overview

Project Summary

- Name: Jungle(JNGL) Token
- Platform: Ethereum
- Address: 0x4C45bbEc2fF7810ef4a77ad7BD4757C446Fe4155
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Jungle(JNGL) Token
Version	v2
Type	Solidity
Date	Oct 09 2023
Logs	Oct 08 2023; Oct 09 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	2
Total informational issues	2
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	Acknowledged
2	Missing event for critical state change	Low	Logging	Acknowledged
3	Could use the 2-step ownership transfer process	Informational	Access Control	Acknowledged
4	Floating compiler version	Informational	Configuration	Acknowledged

2.2 Notable Findings

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

1. Centralization risk

Severity: Low	Category: Centralization
Target: - JungleToken.sol	

Description

The JungleToken contract has a privileged `owner` role. The `owner` can:

- modify the sell tax using setSellTax()
- modify the buy tax using setBuyTax()
- modify the mev tax using setMevTax()
- update the treasury address using setTreasury()
- update the uniswapV2Pair variable using setRule()

It should be noted that there is no upper limit for tax in the setSellTax(), setBuyTax(), and setMevTax() functions. If a malicious owner sets, let's say the buy tax, to 100%, the users will not receive any JNGL tokens when buying JNGL from the `uniswapV2Pair`. The fee will be sent to the `treasury` address, which can be set by the owner.

JungleToken.sol:L677-L688

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 /* amount */
) override internal virtual {

    if (uniswapV2Pair == address(0)) {
        require(from == owner() || to == owner(), "trading is not started");
        return;
    }

}
```

Also note that the _beforeTokenTransfer() hook includes a check: if `uniswapV2Pair` equals address(0), token transfer will revert with an exception for the owner account. This allows a malicious owner to make the JNGL token non-transferable by setting `uniswapV2Pair` to address(0) using setRule().

In summary, should the owner's private key be compromised, the attacker can set taxes to 100% and profit from users' swaps. The attacker can also make the JNGL token non-transferrable, which disrupts the functioning of other contracts integrated with the JNGL token.

Recommendation

The current owner address, 0x52C6a19e6B79CD5D24d32EE26c6da75eEd2e535D, is a

plain EOA account. We recommend transferring the owner role to a multi-sig account with timelock governors for enhanced security.

Additionally, we recommend adding an upper limit for taxes in the setSellTax(), setBuyTax(), and setMevTax() functions. It is also advisable to include a zero-address check for the `uniswapV2Pair` variable in the setRule() function.

Status

This issue has been acknowledged by the team.

2. Missing event for critical state change

Severity: Low	Category: Logging
Target: - JungleToken.sol	

Description

Important parameter or configuration changes should trigger an event to enable tracking off-chain.

However, the setRule(), setTreasury(), setSellTax(), setBuyTax(), and setMevTax() functions alter important states without emitting any events.

Recommendation

It is recommended to design proper events and incorporate them into the aforementioned functions.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

3. Could use the 2-step ownership transfer process

Severity: Informational	Category: Access Control
Target: - JungleToken.sol	

Description

The current ownership transfer process involves the current owner calling transferOwnership(), which writes the new owner's address into the owner's state variable. However, if the nominated EOA account is not a valid account, it is entirely possible the owner may accidentally transfer ownership to an uncontrolled account, breaking all owner-only functions.

It’s recommended to adopt a 2-step process for ownership transfer. In this approach, the owner can designate an address as the owner candidate, but the actual transfer of ownership occurs only when the candidate explicitly accepts the ownership.

Recommendation

Consider using the 2-step process for transferring ownership, e.g. using the Ownable2Step contract from the OpenZeppelin library.

Status

This issue has been acknowledged by the team.

4. Floating compiler version

Severity: Informational	Category: Configuration
Target: - JungleToken.sol	

Description

```
pragma solidity ^0.8.18;
```

The JungleToken contract uses a floating Solidity compiler version, ^0.8.18.

However, we discourage this practice. It's best to deploy contracts with the same compiler version and flags that they have been thoroughly tested with. Locking the compiler version helps to prevent contracts from being accidentally deployed using an outdated compiler version, which could introduce bugs that have a negative impact on the system..

Recommendation

It is recommended to use a locked Solidity compiler version.

For example:

```
pragma solidity 0.8.18;
```

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file from address 0x4C45bbEc2fF7810ef4a77ad7BD4757C446Fe4155:

File	SHA-1 hash
JungleToken.sol	4b9a572847a9b710fac1a658817e550afd4dab5e