Smart contract security audit

RSET

v.1.0

Table of Contents

1.0 Introduction

1.1 Project engagement

During April of 2021, RSET engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. RSET provided CTDSec with access to their code repository and whitepaper.
RSET is a DEX aggregator built in BSC and made to provide the best crypto prices around all BSC DEX and also provide launchpad features to allow ecosystem growth.

1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that RSET team put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the RSET contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

Rset.sol - 4F1562782C3C18832579C41E3776F3AE

Rsetsale.sol - 4D3E0478202BE4B734BBBAEA80DFB822

RsetPrivateSale.sol - BE7FF93F0C993542976495E190E88132

2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

№	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	PASSED
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED

15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED
17	Arithmetic accuracy.	PASSED
18	Design Logic.	PASSED
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	PASSED

3.0 Security Issues

3.1 High severity issues [0]

No high severity issues found.

3.2 Medium severity issues [0]

No medium severity issues found.

3.3 Low severity issues [1]

1. Division before multiply Issue:

There is division before multiply in contract RsetPrivateSale.sol at lines 623 and 644, this can lead to wrong calculations.

Recommendation:

Please multiply before dividing, as below in example:

```
// Token amount per price
uint256 tokensSold = (_value).mul(10 ** tokenDecimals).div(buyPrice);
```

4.0 Summary of the audit

The contract has followed good cybersecurity practices and is safe to deploy.