Smart contract security audit

Evai

v.1.0

Table of Contents

1.0 Introduction **3**

1.1 Project engagement 3

1.2 Disclaimer 3

2.0 Coverage **3**

2.1 Target Code and Revision 3

2.2 Attacks made to the contract 4

3.0 Security Issues **6**

3.1 High severity issues [0] 6

3.2 Medium severity issues [3] 6

3.3 Low severity issues [1 - Fixed] 7

4.0 Summary of the audit **8**

1.0 Introduction

1.1 Project engagement

During April of 2022, Evai 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. Evai provided CTDSec with access to their code repository and whitepaper.

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 Evai 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 Evai 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:

Token-evai.sol [SHA256] -

3c3e3b13dbdbfd9dad832c6debd0aa69d1b409a0227ed129e7847ff1d8834003

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.	MEDIUM ISSUES
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.	MEDIUM ISSUES
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	MEDIUM ISSUES

3.0 Security Issues

3.1 High severity issues [0]

No high severity issues found.

3.2 Medium severity issues [3]

1. Out of gas

The function includeInReward uses the loop to find and remove addresses from the _excluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

Recommendation:

Check that the excluded array lengths are not too big.

2. The function removeallfee may have unexpected behavior

The 'removeallfee' function can have unexpected behavior since only the values of taxfee and liquidityfee are reset but not burning.

```
1136    function removeAllFee() private {
1137        if(_taxFee == 0 && _liquidityFee == 0) return;
1138
1139        _previousTaxFee = _taxFee;
1140        _previousLiquidityFee = _liquidityFee;
1141
1142        _taxFee = 0;
1143        _liquidityFee = 0;
1144    }
```

Recommendation:
Check that the logic is as expected.

3. Centralization

Token.sol

- Owner can change the tax and liquidity fee.
- Owner can change the maximum transaction amount.
- Owner can be excluded from the fee.
- Owner can lock and unlock. By the way, using these functions the owner could leave as owner even after the ownership was renounced.
- Owner can change the Development and Marketing wallets.
- Owner can change the lp reward from liquidity percent.

Recommendation:

-Add a time lock on privileged operations.

-Use a multisig wallet to prevent SPOF on the Private Key.

-Introduce DAO mechanism for owner functions (will add transparency and user involvement).

3.3 Low severity issues [1 - Fixed]

1. Testing variables

The project contains test strings assigned to some parameters such as name and symbol.

```
738     string private _name = "Demo Token";
739     string private _symbol = "DTK";
```

Recommendation:

Review all the variables previously put into production.

4.0 Summary of the audit

Contract is safe to be deployed, we recommend that the team make use of a multisignature wallet for the owner role.