慢雾科技
SLOWMIST

智能合约安全审计报告

[2021]

慢雾安全团队于2021.09.07，收到Octopus Network团队对Octopus Network Token智能合约安全审计的申请，如下为本次智能合约安全审计细节及结果：

Token 名称：

Octopus Network Token

合约地址：

https://github.com/octopus-network/oct-token-eth/tree/main/contracts

Commit：

5c7b93fe00dd22b8d67047f136c399915ed3e514

本次审计项及结果：

（其他未知安全漏洞不包含在本次审计责任范围）

序号	审计类别	审计结果
1	重放攻击	通过
2	拒绝服务攻击	通过
3	条件竞争攻击	通过
4	权限控制攻击	通过
5	整数上溢/下溢攻击	通过
6	Gas优化设计	通过
7	业务逻辑缺陷审计	通过
8	未声明的存储指针	通过
9	算术精度误差	通过
10	假充值漏洞	通过

序号	审计类别	审计结果
11	恶意 Event 事件审计	通过
12	变量声明及作用域审计	通过
13	安全设计审计	通过

审计结果：通过

审计编号：0x002109080002

审计日期：2021.09.07 - 2021.09.08

审计团队：SlowMist Security Team

备注：审计意见及建议见代码注释 //SlowMist//……

总结： 此为代币 (token) 合约，包含时间锁 (Timelock) 部分。合约的代币总量不可变。使用了 SafeMath 安全模块，值得称赞的做法。合约不存在溢出、条件竞争问题。合约存在权限过大的风险问题。

在审计过程中，我们发现如下信息：

1. SupervisedTimelock 合约中 owner 角色可以通过调用 terminate 函数提取合约中所有剩余的代币，并且合约的 _isTerminated 状态将被设为 true 不会再改变。

合约源代码如下：

OctToken.sol

```
// SPDX-License-Identifier: GPL-3.0
//SlowMist// 合约不存在溢出、条件竞争问题
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract OctToken is ERC20, Ownable {
    // Total supply: 100 million
    uint256 private constant TOTAL_SUPPLY = 100000000;
    /**
     * @dev Initializes the contract, mint total supply to the deployer (owner).
     */
    constructor() ERC20("Octopus Network Token", "OCT") {
        _mint(msg.sender, TOTAL_SUPPLY * 10**(uint256(decimals())));
    }
}
```

UnsupervisedTimelock.sol

```
// SPDX-License-Identifier: GPL-3.0
//SlowMist// 合约不存在溢出、条件竞争问题
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

/**
 * @dev A token holder contract that will allow a beneficiary to withdraw the
 * tokens after a given release time.
 */
contract UnsupervisedTimelock {
    using SafeERC20 for IERC20;
    // Seconds of a day
    uint256 private constant SECONDS_OF_A_DAY = 86400;
    // beneficiary of tokens after they are released
    address private immutable _beneficiary;
    // The start timestamp of token release period.
    //
    // Before this time, the beneficiary can NOT withdraw any token from this
contract.
    uint256 private immutable _releaseStartTime;
    // The days that the timelock will last.
    uint256 private immutable _daysOfTimelock;
    // The OctToken contract
    IERC20 private immutable _token;
    // Total balance of benefit
    uint256 private immutable _totalBenefit;
    // The amount of withdrawed balance of the beneficiary.
    //
    // This value will be updated on each withdraw operation.
    uint256 private _withdrawedBalance;

    event BenefitWithdrawed(address indexed beneficiary, uint256 amount);
```

```
constructor(
    IERC20 token_,
    address beneficiary_,
    uint256 releaseStartTime_,
    uint256 daysOfTimelock_,
    uint256 totalBenefit_
) {
    _token = token_;
    _beneficiary = beneficiary_;
    _releaseStartTime =
        releaseStartTime_ -
        (releaseStartTime_ % SECONDS_OF_A_DAY);
    require(
        releaseStartTime_ -
            (releaseStartTime_ % SECONDS_OF_A_DAY) +
            daysOfTimelock_ *
            SECONDS_OF_A_DAY >
            block.timestamp,
        "UnsupervisedTimelock: release end time is before current time"
    );
    _daysOfTimelock = daysOfTimelock_;
    _totalBenefit = totalBenefit_;
    _withdrawedBalance = 0;
}

/**
 * @return the token being held.
 */
function token() public view returns (IERC20) {
    return _token;
}

/**
 * @return the total balance of benefit
 */
function totalBenefit() public view returns (uint256) {
    return _totalBenefit;
}

/**
 * @return the balance to release for the beneficiary at the moment
 */
function releasedBalance() public view returns (uint256) {
    if (block.timestamp <= _releaseStartTime) return 0;
    if (
        block.timestamp >
        _releaseStartTime + SECONDS_OF_A_DAY * _daysOfTimelock
    ) {
        return _totalBenefit;
    }
    uint256 passedDays = (block.timestamp - _releaseStartTime) /
        SECONDS_OF_A_DAY;
    return (_totalBenefit * passedDays) / _daysOfTimelock;
}

/**
 * @return the unreleased balance of the beneficiary at the moment
 */
function unreleasedBalance() public view returns (uint256) {
    return _totalBenefit - releasedBalance();
}

/**
 * @return the withdrawed balance of beneficiary
 */
function withdrawedBalance() public view returns (uint256) {
    return _withdrawedBalance;
}

/**
 * @notice Withdraws tokens to beneficiary
 */
function withdraw() public {
    uint256 balanceShouldBeReleased = releasedBalance();
    require(
        balanceShouldBeReleased > _withdrawedBalance,
        "UnsupervisedTimelock: no more benefit can be withdrawed now"
    );
    uint256 balanceShouldBeTransfered = balanceShouldBeReleased -
        _withdrawedBalance;
    require(
        token().balanceOf(address(this)) >= balanceShouldBeTransfered,
        "UnsupervisedTimelock: deposited balance is not enough"
    );

    _withdrawedBalance = balanceShouldBeReleased;

    token().safeTransfer(_beneficiary, balanceShouldBeTransfered);

    emit BenefitWithdrawed(_beneficiary, balanceShouldBeTransfered);
}
}
```

SupervisedTimelock.sol

```
// SPDX-License-Identifier: GPL-3.0
//SlowMist// 合约不存在溢出、条件竞争问题
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

/**
 * @dev A token holder contract that will allow a beneficiary to withdraw the
 * tokens after a given release time.
 */
contract SupervisedTimelock is Ownable {
    using SafeERC20 for IERC20;
    // Seconds of a day
    uint256 private constant SECONDS_OF_A_DAY = 86400;
    // The OctToken contract
    IERC20 private immutable _token;
    // beneficiary of tokens after they are released
    address private immutable _beneficiary;
    // The start timestamp of token release period.
    //
    // Before this time, the beneficiary can NOT withdraw any token from this
contract.
    uint256 private immutable _releaseStartTime;
    // The end timestamp of token release period.
    //
    // After this time, the beneficiary can withdraw all amount of benefit.
    uint256 private _releaseEndTime;
    // Total balance of benefit
    uint256 private _totalBenefit;
    // The amount of withdrawed balance of the beneficiary.
    //
    // This value will be updated on each withdraw operation.
    uint256 private _withdrawedBalance;
    // The flag of whether this contract is terminated.
    bool private _isTerminated;

    event BenefitWithdrawed(address indexed beneficiary, uint256 amount);
    event ContractIsTerminated(address indexed beneficiary, uint256 amount);
```

```
constructor(
    IERC20 token_,
    address beneficiary_,
    uint256 releaseStartTime_,
    uint256 daysOfTimelock_,
    uint256 totalBenefit_
) {
    _token = token_;
    _beneficiary = beneficiary_;
    releaseStartTime_ -= (releaseStartTime_ % SECONDS_OF_A_DAY);
    _releaseStartTime = releaseStartTime_;
    _releaseEndTime =
        releaseStartTime_ +
        daysOfTimelock_ *
        SECONDS_OF_A_DAY;
    require(
        _releaseEndTime > block.timestamp,
        "SupervisedTimelock: release end time is before current time"
    );
    _totalBenefit = totalBenefit_;
    _withdrawedBalance = 0;
    _isTerminated = false;
}

/**
 * @dev Throws if called by any account other than the owner.
 */
modifier isNotTerminated() {
    require(
        _isTerminated == false,
        "SupervisedTimelock: this contract is terminated"
    );
    _;
}

/**
 * @return the token being held.
 */
function token() public view returns (IERC20) {
    return _token;
}

/**
 * @return the amount of total benefit
 */
function totalBenefit() public view returns (uint256) {
    return _totalBenefit;
}

/**
 * @return the balance which can be withdrawed at the moment
 */
function releasedBalance() public view returns (uint256) {
    if (block.timestamp <= _releaseStartTime) return 0;
    if (block.timestamp > _releaseEndTime) {
        return _totalBenefit;
    }
    uint256 passedDays = (block.timestamp - _releaseStartTime) /
        SECONDS_OF_A_DAY;
    uint256 totalDays = (_releaseEndTime - _releaseStartTime) /
        SECONDS_OF_A_DAY;
    return (_totalBenefit * passedDays) / totalDays;
}

/**
 * @return the unreleased balance at the moment
 */
function unreleasedBalance() public view returns (uint256) {
    return _totalBenefit - releasedBalance();
}

/**
 * @return the withdrawed balance at the moment
 */
function withdrawedBalance() public view returns (uint256) {
    return _withdrawedBalance;
}

/**
 * @notice Withdraws tokens to beneficiary
 */
function withdraw() public {
    require(
        releasedBalance() > _withdrawedBalance,
        "SupervisedTimelock: no more benefit to withdraw"
    );
    uint256 amount = releasedBalance() - _withdrawedBalance;
    require(
        token().balanceOf(address(this)) >= amount,
            "SupervisedTimelock: deposited amount is not enough"
        );

        _withdrawedBalance += amount;
        token().safeTransfer(_beneficiary, amount);

        emit BenefitWithdrawed(_beneficiary, amount);
    }

    /**
     * @notice Teminate this contract and withdraw all amount of unreleased balance
to the owner.
     * After the contract is terminated, the beneficiary can still withdraw all
amount of
     * released balance.
     */
    //SlowMist// 合约 owner 角色可以通过调用 terminate 函数提取合约中所有剩余的代币，并且合约的
_isTerminated 状态将被设为 true 不会再改变。
    function terminate() public onlyOwner isNotTerminated {
        _totalBenefit = releasedBalance();
        _releaseEndTime =
            block.timestamp -
            (block.timestamp % SECONDS_OF_A_DAY);
        _isTerminated = true;

        uint256 amountToWithdraw = token().balanceOf(address(this)) -
            (_totalBenefit - _withdrawedBalance);
        token().safeTransfer(owner(), amountToWithdraw);

        emit ContractIsTerminated(owner(), amountToWithdraw);
    }
}
```

声明

厦门慢雾科技有限公司(下文简称 ”慢雾”) 仅就本报告出具前项目方已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后项目方发生或存在的未知漏洞及安全事件，慢雾无法判断其安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料(简称“已提供资料”)。慢雾假设: 已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任，慢雾仅对该项目的安全情况进行约定内的安全审计并出具了本报告，慢雾不对该项目背景及其他情况进行负责。

慢雾科技
SLOWMIST

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号