Smart Contract Security Audit Report

The SlowMist Security Team received the Elevate team's application for smart contract security audit of the ELE on Mar. 04, 2021. The following are the details and results of this smart contract security audit:

Token name :

ELE

The Contract address :

https://github.com/elevatedefi/elevate-contracts/blob/main/ELEVATE.sol

commit: 456a51ed87384e7e383f266f41136528749e81dd

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

No.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	-	Passed
2	Race Conditions Audit	-	Passed
3	Authority Control Audit	Permission vulnerability audit	Passed
		Excessive authority audit	Passed
4	Safety Design Audit	Zeppelin module safe use	Passed
		Compiler version security	Passed
		Hard-coded address security	Passed
		Fallback function safe use	Passed
		Show coding security	Passed
		Function return value security	Passed
		Call function security	Passed
5	Denial of Service Audit	-	Passed
6	Gas Optimization Audit	-	Passed
7	Design Logic Audit	-	Passed
8	“False top-up” vulnerability Audit	-	Passed
9	Malicious Event Log Audit	-	Passed

10	Scoping and Declarations Audit	-	Passed
11	Replay Attack Audit	ECDSA's Signature Replay Audit	Passed
12	Uninitialized Storage Pointers Audit	-	Passed
13	Arithmetic Accuracy Deviation Audit	-	Passed

Audit Result : Passed

Audit Number : 0X002103090001

Audit Date : Mar. 09, 2021

Audit Team : SlowMist Security Team

Summary: This is a token contract that does not contain the tokenVault section. The total amount of tokens in the contract remains unchanged. OpenZeppelin's SafeMath security module is used, which is a recommend approach. The contract does not have the Overflow and the Race Conditions issue. During the audit, we found the following information:

If ELE tokens add liquidity to a DEX similar to Uniswap, then when ELE is inflated, the number of ELE tokens in the pool will also automatically increase. At this time, users can withdraw the added tokens through the skim function of the pair contract. After communicating with the project party, the project party stated: The excludeAccount function is used on the Uniswap pool address to prevent the pool from earning inflation.

The source code:

```
/*
 * Copyright © 2021 elevatedefi.io. ALL RIGHTS RESERVED.
 */
```

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

```
pragma solidity ^0.6.2;

import "openzeppelin-solidity/contracts/GSN/Context.sol";
import "openzeppelin-solidity/contracts/token/ERC20/IERC20.sol";
import "openzeppelin-solidity/contracts/math/SafeMath.sol";
import "openzeppelin-solidity/contracts/utils/Address.sol";
import "openzeppelin-solidity/contracts/access/Ownable.sol";

contract ELEVATE is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;

    mapping (address => bool) private _isExcluded;
    address[] public _excluded;
    uint256 private constant _max_excluded = 10;

    uint256 private constant MAX = ~uint256(0);
    uint256 private constant _tInitial = 10 * 10**6 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tInitial));
    uint256 private _tFeeTotal;
    uint256 private _tBurned;
    uint256 private _startTime;
    address private multisig = address(0xD4Cb1274d46c96B576Dd599e370197e992ff8753);

    string private _name = 'Elevate';
    string private _symbol = 'ELE';
    uint8 private _decimals = 9;

    constructor () public {
        _startTime = now;
        _rOwned[_msgSender()] = _rTotal;
```

```
        emit Transfer(address(0), _msgSender(), _tInitial);
    }

    function name() public view returns (string memory) {
        return _name;
    }

    function symbol() public view returns (string memory) {
        return _symbol;
    }

    function decimals() public view returns (uint8) {
        return _decimals;
    }

    function totalSupply() public view override returns (uint256) {
        return _getTotalSupply(_startTime);
    }

    function balanceOf(address account) public view override returns (uint256) {
        if (_isExcluded[account]) return _tOwned[account];
        return tokenFromReflection(_rOwned[account]);
    }

    function transfer(address recipient, uint256 amount) public override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function allowance(address owner, address spender) public view override returns (uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) public override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
```

```
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
        return true;
    }

    function isExcluded(address account) public view returns (bool) {
        return _isExcluded[account];
    }

    function totalFees() public view returns (uint256) {
        return _tFeeTotal;
    }

    function totalBurned() public view returns (uint256) {
        return _tBurned;
    }

    function startedAt() public view returns (uint256) {
        return _startTime;
    }

    function isGenesis() public view returns(bool) {
        return _isGenesis(_startTime, now);
    }

    function reflect(uint256 tAmount, bool includeBurn) public {
        address sender = _msgSender();
```

```
        require(!_isExcluded[sender], "Excluded addresses cannot call this function");
        if (!includeBurn) {
            (uint256 rAmount,,,,,,) = _getValues(tAmount);
            _rOwned[sender] = _rOwned[sender].sub(rAmount);
            _reflectFee(rAmount, tAmount, 0);
        } else {
            (uint256 rAmount,,,,,,) = _getValues(tAmount);
            _rOwned[sender] = _rOwned[sender].sub(rAmount);
            uint256 tBurn = tAmount.div(2);
            uint256 tFee = tAmount.sub(tBurn);
            _reflectFee(rAmount, tFee, tBurn);
        }
    }

    function burn(uint256 tAmount) public {
        address sender = _msgSender();
        require(!_isExcluded[sender], "Excluded addresses cannot call this function");
        (uint256 rAmount,,,,,,) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _reflectFee(rAmount, 0, tAmount);
    }

    function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
        require(tAmount <= _getTotalSupply(_startTime), "Amount must be less than supply");
        if (!deductTransferFee) {
            (uint256 rAmount,,,,,,) = _getValues(tAmount);
            return rAmount;
        } else {
            (,uint256 rTransferAmount,,,,,) = _getValues(tAmount);
            return rTransferAmount;
        }
    }

    function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
        require(rAmount <= _rTotal, "Amount must be less than total reflections");
        uint256 currentRate =  _getRate();
        return rAmount.div(currentRate);
    }

    function excludeAccount(address account) external onlyOwner() {
```

```
        require(!_isExcluded[account], "Account is already excluded");
        require(_excluded.length < _max_excluded, "Max number of accounts excluded");
        require(account != multisig, "Multisig cannot be excluded");
        if(_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    }

    function includeAccount(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is not excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                uint256 _rBalance = reflectionFromToken(_tOwned[account], false);
                _rTotal = _rTotal.sub(_rOwned[account].sub(_rBalance));
                _rOwned[account] = _rBalance;
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded[i] = _excluded[_excluded.length - 1];
                _excluded.pop();
                break;
            }
        }
    }

    function _approve(address owner, address spender, uint256 amount) private {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    function _transfer(address sender, address recipient, uint256 amount) private {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");
        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient, amount);
```

```
    } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
        _transferToExcluded(sender, recipient, amount);
    } else if (_isExcluded[sender] && _isExcluded[recipient]) {
        _transferBothExcluded(sender, recipient, amount);
    } else {
        _transferStandard(sender, recipient, amount);
    }
}

function _transferStandard(address sender, address recipient, uint256 tAmount) private {
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount,,
     uint256 tFeeHalf, uint256 currentRate) = _getValues(tAmount);
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    _distributeFee(sender, rFee, tFeeHalf, currentRate);
    emit Transfer(sender, recipient, tTransferAmount);
}

function _transferToExcluded(address sender, address recipient, uint256 tAmount) private {
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount,,
     uint256 tFeeHalf, uint256 currentRate) = _getValues(tAmount);
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
    _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    _distributeFee(sender, rFee, tFeeHalf, currentRate);
    emit Transfer(sender, recipient, tTransferAmount);
}

function _transferFromExcluded(address sender, address recipient, uint256 tAmount) private {
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount,,
     uint256 tFeeHalf, uint256 currentRate) = _getValues(tAmount);
    _tOwned[sender] = _tOwned[sender].sub(tAmount);
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    _distributeFee(sender, rFee, tFeeHalf, currentRate);
    emit Transfer(sender, recipient, tTransferAmount);
}

function _transferBothExcluded(address sender, address recipient, uint256 tAmount) private {
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount,,
```

```
        uint256 tFeeHalf, uint256 currentRate) = _getValues(tAmount);
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _distributeFee(sender, rFee, tFeeHalf, currentRate);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _distributeFee(address sender, uint256 rFee, uint256 tFeeHalf, uint256 currentRate) private {
        if (_isGenesis(_startTime, now)) {
            (uint256 rNewFee, uint256 tBurn, uint256 rGenesisCut, uint256 tGenesisCut) = _getGenesisValues(rFee, tFeeHalf,
currentRate);
            _rOwned[multisig] =  _rOwned[multisig].add(rGenesisCut);
            _reflectFee(rNewFee, tFeeHalf, tBurn);
            emit Transfer(sender, multisig, tGenesisCut);
        } else {
            _reflectFee(rFee, tFeeHalf, tFeeHalf);
        }
    }

    function _reflectFee(uint256 rFee, uint256 tFee, uint256 tBurn) private {
        _rTotal = _rTotal.sub(rFee);
        _tFeeTotal = _tFeeTotal.add(tFee);
        _tBurned = _tBurned.add(tBurn);
    }

    function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, uint256, uint256, uint256, uint256)
{
        (uint256 tTransferAmount, uint256 tFee, uint256 tFeeHalf) = _getTValues(tAmount);
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmount, tFee, currentRate);
        return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tFeeHalf, currentRate);
    }

    function _getTValues(uint256 tAmount) private pure returns (uint256, uint256, uint256) {
        //SlowMist// When calculating tFee, divide first and then multiply, which will result in loss of
accuracy
        uint256 tFeeHalf = tAmount.div(200);
```

```
        uint256 tFee = tFeeHalf.mul(2);
        uint256 tTransferAmount = tAmount.sub(tFee);
        return (tTransferAmount, tFee, tFeeHalf);
    }

    function _getRValues(uint256 tAmount, uint256 tFee, uint256 currentRate) private pure returns (uint256, uint256, uint256)
{
        uint256 rAmount = tAmount.mul(currentRate);
        uint256 rFee = tFee.mul(currentRate);
        uint256 rTransferAmount = rAmount.sub(rFee);
        return (rAmount, rTransferAmount, rFee);
    }

    function _getGenesisValues(uint256 rFee, uint256 tFeeHalf, uint256 currentRate) private pure returns (uint256, uint256,
uint256, uint256) {
        //SlowMist// When calculating rGenesisCut, divide first and then multiply, which will result in
loss of accuracy
        uint256 tGenesisCut = tFeeHalf.div(2);
        uint256 tBurn = tFeeHalf.sub(tGenesisCut);
        uint256 rGenesisCut = tGenesisCut.mul(currentRate);
        uint256 rNewFee = rFee.sub(rGenesisCut);
        return (rNewFee, tBurn, rGenesisCut, tGenesisCut);
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply.div(tSupply);
    }

    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tTotal = _getTotalSupply(_startTime);
        uint256 tSupply = tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, tTotal);
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        return (rSupply, tSupply);
```

```
    }

    function _getTotalSupply(uint256 startTime) private view returns(uint256) {
        return _getInflatedSupply(_tInitial, startTime, now).sub(_tBurned);
    }

    function _getInflatedSupply(uint256 initialSupply, uint256 startTime, uint256 currentTime) private pure returns(uint256) {
        return initialSupply.add(_getElapsedSeconds(startTime, currentTime).mul(158548959));
    }

    function _getElapsedSeconds(uint256 startTime, uint256 currentTime) private pure returns(uint256) {
        return currentTime.sub(startTime);
    }

    function _isGenesis(uint256 startTime, uint256 currentTime) private pure returns(bool) {
        return currentTime.sub(startTime) < 15 days;
    }
}
```

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist