FDU

Intelligent contract audit report

Singapore GreenLand digital security technology co., LTD

Contract name:

FDU DeFi

Contract address:

0x29bb5dc47481341bad5a6991db7065e8c397f652

Contract link:

https://etherscan.io/address/0x29bb5dc47481341bad5a6991db7065e8c397f652

Contract audit start date:

2020.11.21

Contract audit completion date:

2020.11.26

Audit results:

Pass (Excellent)

Audit team:

Singapore GreenLand digital security technology co., LTD

Audit type and results:

	Audit type	**Audit sub-items**	**Audit results**
1	Code specification audit	ERC20 Token Standard Specification Audit	Pass
		Compiler version security audit	Pass
		Visibility specification audit	Pass
		gas consumption audit	Pass
		SafeMath function audit	Pass
		fallback function usage audit	Pass
		tx.origin usage audit	Pass
		Deprecated item audit	Pass
		Redundant code audit	Pass

		Variable coverage audit	Pass
2	Function call audit	Function call permission audit	Pass
		call/delegatecall security audit	Pass
		Return value security audit	Pass
		Self-destruct function security audit	Pass
3	Business security audit	Owner permission audit	Pass
		Business logic audit	Pass
		Business realization audit	Pass
4	Integer overflow audit	-	Pass
5	Reentrant attack audit	-	Pass
6	Abnormal reachability audit	-	Pass
7	Transaction order depends on audit	-	Pass
8	Block parameter dependency audit	-	Pass
9	Pseudo-random number generation audit	-	Pass
10	Denial of service attack audit	-	Pass
11	Token lock-up audit	-	Pass
12	Fake recharge audit	-	Pass
13	event security audit	-	Pass

Disclaimer: This audit is only conducted for the audit type specified in this report and the scope of the audit type specified in the result table, and other unknown security vulnerabilities are not within the scope of this audit. GreenLand only issues this report based on existing or occurring attacks or vulnerabilities before the issuance of this report. For new attacks or vulnerabilities that exist or occur in the future, GreenLand cannot determine its possible impact on the security status of smart contracts, and is not responsible for them. . The security audit analysis and other content made in this report are based only on the documents and information that the contract provider has provided to GreenLand before the issuance of this report, and there are no preconditions for any missing, tampered, deleted, or concealed documents and information. If the documents and materials provided are missing, tampered with, deleted, concealed or reflected and do not match the actual situation, or if the documents and materials

provided are changed after the issuance of this report, GreenLand is The resulting losses and adverse effects shall not be liable. The audit report issued by GreenLand is based on the documents and information provided by the contract provider and relying on the technology currently mastered by GreenLand. Due to the technical limitations of any organization, the audit report issued by GreenLand still cannot fully detect all risks. GreenLand shall not be liable for any losses arising therefrom.

The final interpretation of this statement belongs to GreenLand.

Description of audit results:

The company uses formal verification, static analysis, dynamic analysis, typical case testing, and manual review to conduct multi-dimensional and comprehensive security audits on the code standardization, security, and business logic of the smart contract FDU DeFi Coin. After auditing, the FDU DeFi Coin contract passed all inspection items, and the contract audit result was passed (excellent), and the contract can be used normally. The following is the basic information of this contract.

1. Basic token information

Token name	FDU DeFi Coin
Token symbol	FDU DeFi Coin
decimals	8
totalSupply	9,000,000 (Can be destroyed)
Token type	ERC20

2. Contract source code

```
/**
*Submitted for verification at Etherscan.io on 2020-11-20
*FDU DEFI
*www.fdu.life
*/

pragma solidity ^0.4.16;

interface tokenRecipient { function receiveApproval(address _from, uint256 _value, address _token, bytes
_extraData) public; }

//FDU DEFI

contract FDUtoken {
    string public name;
    string public symbol;
    uint8 public decimals = 8;
    uint256 public totalSupply;

    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Burn(address indexed from, uint256 value);

    event Approval(address indexed _owner, address indexed _spender, uint256 _value);


    function FDUtoken(uint256 initialSupply, string tokenName, string tokenSymbol) public {
        totalSupply = initialSupply * 10 ** uint256(decimals);
        balanceOf[msg.sender] = totalSupply;
        name = tokenName;
        symbol = tokenSymbol;
    }


    function _transfer(address _from, address _to, uint _value) internal {
        require(_to != 0x0);
        require(balanceOf[_from] >= _value);
        require(balanceOf[_to] + _value > balanceOf[_to]);
        uint previousBalances = balanceOf[_from] + balanceOf[_to];
```

```
        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;
        Transfer(_from, _to, _value);
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
    }

    function transfer(address _to, uint256 _value) public returns (bool) {
        _transfer(msg.sender, _to, _value);
        return true;
    }

    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
        require(_value <= allowance[_from][msg.sender]);
        allowance[_from][msg.sender] -= _value;
        _transfer(_from, _to, _value);
        return true;
    }

    function approve(address _spender, uint256 _value) public
        returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function approveAndCall(address _spender, uint256 _value, bytes _extraData) public returns (bool success)
{
        tokenRecipient spender = tokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this, _extraData);
            return true;
        }
    }

    function burn(uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);
        balanceOf[msg.sender] -= _value;
        totalSupply -= _value;
        Burn(msg.sender, _value);
        return true;
    }

    function burnFrom(address _from, uint256 _value) public returns (bool success) {
```

```
        require(balanceOf[_from] >= _value);
        require(_value <= allowance[_from][msg.sender]);
        balanceOf[_from] -= _value;
        allowance[_from][msg.sender] -= _value;
        totalSupply -= _value;
        Burn(_from, _value);
        return true;
    }
}
```