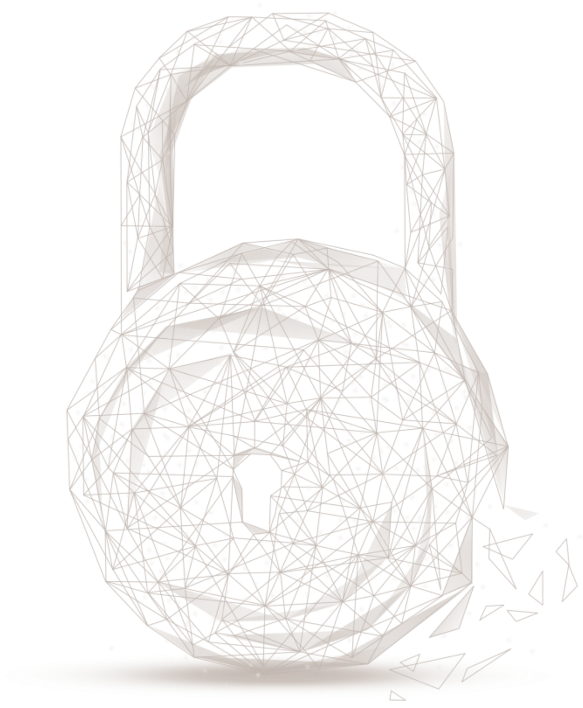
BEOSIN
Blockchain Security

# Smart contract security audit report



**Audit Number：202103221455**

**Smart Contract Name：**

Bytus (BYTS)

**Smart Contract Address：**

0x87F14E9460ceCb789F1B125b2E3e353Ff8ed6fcd

**Smart Contract Address Link：**

https://etherscan.io/address/0x87F14E9460ceCb789F1B125b2E3e353Ff8ed6fcd#code

**Start Date：2021.03.18**

**Completion Date：2021.03.22**

**Overall Result：Pass**

**Audit Team: Beosin (Chengdu LianAn) Technology Co. Ltd.**

**Audit Categories and Results:**

| No. | Categories | Subitems | Results |
|---|---|---|---|
| 1 | Coding Conventions | ERC20 Token Standards | Fail |
| | | Compiler Version Security | Pass |
| | | Visibility Specifiers | Pass |
| | | Gas Consumption | Pass |
| | | SafeMath Features | Pass |
| | | Fallback Usage | Pass |
| | | tx.origin Usage | Pass |
| | | Deprecated Items | Pass |
| | | Redundant Code | Pass |
| | | Overriding Variables | Pass |
| 2 | Function Call Audit | Authorization of Function Call | Pass |
| | | Low-level Function (call/delegatecall) Security | Pass |
| | | Returned Value Security | Pass |
| | | selfdestruct Function Security | Pass |
| 3 | Business Security | Access Control of Owner | Pass |

|  |  |  |  |
|---|---|---|---|
|  |  | Business Logics | Pass |
|  |  | Business Implementations | Pass |
| 4 | Integer Overflow/Underflow | - | Pass |
| 5 | Reentrancy | - | Pass |
| 6 | Exceptional Reachable State | - | Pass |
| 7 | Transaction-Ordering Dependence | - | Pass |
| 8 | Block Properties Dependence | - | Pass |
| 9 | Pseudo-random Number Generator (PRNG) | - | Pass |
| 10 | DoS (Denial of Service) | - | Pass |
| 11 | Token Vesting Implementation | - | N/A |
| 12 | Fake Deposit | - | Pass |
| 13 | event security | - | Pass |

Note: Audit results and suggestions in code comments

Disclaimer: This audit is only applied to the type of auditing specified in this report and the scope of given in the results table. Other unknown security vulnerabilities are beyond auditing responsibility. Beosin (Chengdu LianAn) Technology only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Beosin (Chengdu LianAn) Technology lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The security audit analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Beosin (Chengdu LianAn) Technology before the issuance of this report, and the contract provider warrants that there are no missing, tampered, deleted; if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Beosin (Chengdu LianAn) Technology assumes no responsibility for the resulting loss or adverse effects. The audit report issued by Beosin (Chengdu LianAn) Technology is based on the documents and materials provided by the contract provider, and relies on the technology currently possessed by Beosin (Chengdu LianAn). Due to the technical limitations of any organization, this report conducted by Beosin (Chengdu LianAn) still has the possibility that the entire risk cannot be completely detected. Beosin (Chengdu LianAn) disclaims any liability for the resulting losses.

The final interpretation of this statement belongs to Beosin (Chengdu LianAn).

**Audit Results Explained:**

Beosin (Chengdu LianAn) Technology has used several methods including Formal Verification, Static Analysis, Typical Case Testing and Manual Review to audit three major aspects of smart contract BYTS, including Coding Standards, Security, and Business Logic. **BYTS contract didn’t pass all audit items. It failed at 'ERC20 Token Standard', which could not affect the usual transaction of token. The overall result is Pass. The smart contract is able to function properly.** Please find below the basic information of the smart contract:

1. Basic Token Information

| Token name | Bytus |
|---|---|
| Token symbol | BYTS |
| decimals | 3 |
| totalSupply | 66 million (Burnable) |
| Token type | ERC20 |

Table 1 – Basic Token Information

2. Token Vesting Information

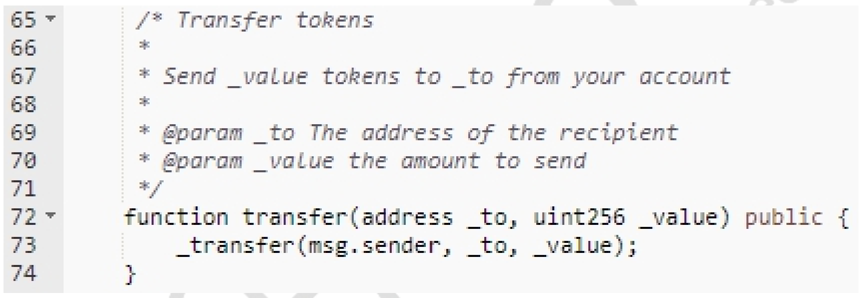
N/A

## Detailed explanations of the 'Fails' in Results

1. ERC20 Token Standard

(1) The transfer function lacks a return value

- Description: As shown in Figure 1, the *transfer* function of this contract lacks a return value. According to the ERC20 Token Standard, the *transfer* function should return a Boolean value. If the external contracts (compiled with version 0.4.22 or above) invoke this function with the ABI which according to ERC20 Token Standard (return value), the invocation will be revert.



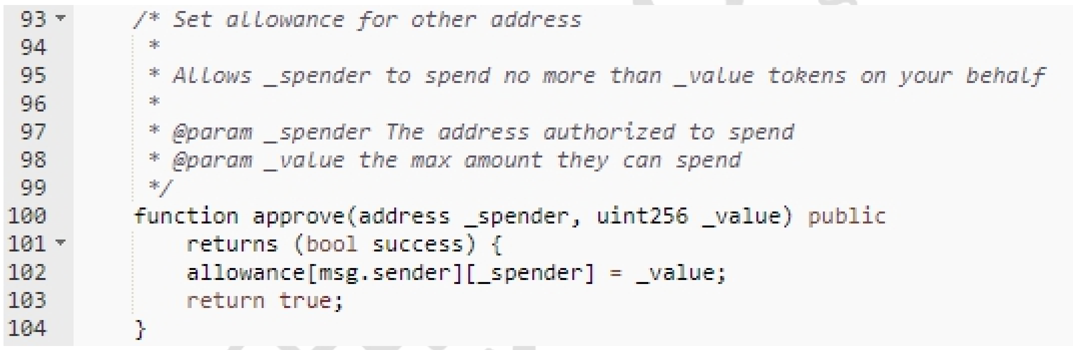
```
/* Transfer tokens
 *
 * Send _value tokens to _to from your account
 *
 * @param _to The address of the recipient
 * @param _value the amount to send
 */
function transfer(address _to, uint256 _value) public {
    _transfer(msg.sender, _to, _value);
}
```

Figure 1 The source code of function 'transfer'

- Safety suggestion: Add the corresponding return value in the 'transfer' function.

(2) Lack of the event 'Approval'

- Description: As shown in Figure 2, the event 'Approval' is not declared in this contract, and the *approve* function does not trigger this event, it does not conform to the ERC20 Token Standard.



```
    /* Set allowance for other address
     *
     * Allows _spender to spend no more than _value tokens on your behalf
     *
     * @param _spender The address authorized to spend
     * @param _value the max amount they can spend
     */
    function approve(address _spender, uint256 _value) public
        returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        return true;
    }
```

Figure 2 The source code of function 'approve'

➢ Safety suggestion: Declare the event 'Approval' in the contract and trigger it in *approve* function.

**Audited Source Code with Comments**

```
/**
 *Submitted for verification at Etherscan.io on 2020-07-23
*/

pragma solidity ^0.4.15;    // Beosin (Chengdu LianAn) // It is recommended to fix the compiler version.
// Beosin (Chengdu LianAn) // Interface, define the function 'receiveApproval' to receive the allowance.
interface tokenRecipient { function receiveApproval(address _from, uint256 _value, address _token, bytes
_extraData) external; }

contract BytusERC20 {
        // Public variables of the token
        string public name;    // Beosin (Chengdu LianAn) // Declare the variable 'name' to store the name of
token.
        string public symbol;    // Beosin (Chengdu LianAn) // Declare the variable 'symbol' to store the symbol
of token.
        uint8 public decimals = 3;    // Beosin (Chengdu LianAn) // Declare the variable 'decimals' to store the
decimals of token.
        // 3 decimals is the strongly suggested default, avoid changing it
        uint256 public totalSupply;    // Beosin (Chengdu LianAn) // Declare the variable 'totalSupply' to store the
total supply of token.

        // This creates an array with all balances
        mapping (address => uint256) public balanceOf;    // Beosin (Chengdu LianAn) // Declare the mapping
variable 'balanceOf' for storing the token balance of corresponding address.
        mapping (address => mapping (address => uint256)) public allowance;    // Beosin (Chengdu LianAn) //
Declare the mapping variable 'allowance' for storing the allowance between two addresses.

        // This generates a public event on the blockchain that will notify clients
        event Transfer(address indexed from, address indexed to, uint256 value);    // Beosin (Chengdu LianAn) //
Declare the event 'Transfer'.

        // This notifies clients about the amount burnt
```

```
    event Burn(address indexed from, uint256 value);    // Beosin (Chengdu LianAn) // Declare the event
'Burn'.


    /* Constructor function
     *
     * Initializes contract with initial supply tokens to the creator of the contract
     */
    constructor (
        uint256 initialSupply,
        string tokenName,
        string tokenSymbol
  ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
// Beosin (Chengdu LianAn) // Initialize the total token supply.
        balanceOf[msg.sender] = totalSupply; // Give the creator all initial tokens    // Beosin (Chengdu
LianAn) // Send all initial tokens to the deployer address.
        name = tokenName; // Set the name for display purposes    // Beosin (Chengdu LianAn) // Initialize the
token name.
        symbol = tokenSymbol; // Set the symbol for display purposes     // Beosin (Chengdu LianAn) //
Initialize the token symbol.
    }


     /* Internal transfer, only can be called by this contract
     */
    function _transfer(address _from, address _to, uint _value) internal {
        // Prevent transfer to 0x0 address. Use burn() instead
        require(_to != 0x0);    // Beosin (Chengdu LianAn) // The non-zero address check for '_to'.
        // Check if the sender has enough
        require(balanceOf[_from] >= _value);    // Beosin (Chengdu LianAn) // The balance check, require
that the transfer value should be no greater than the balance of '_from'.
        // Check for overflows
        require(balanceOf[_to] + _value >= balanceOf[_to]);    // Beosin (Chengdu LianAn) // Overflow check.
        // Save this for an assertion in the future
        uint previousBalances = balanceOf[_from] + balanceOf[_to];    // Beosin (Chengdu LianAn) // Declare
the local variable 'previousBalances ' for storing the token balance sum of addresses '_from' and '_to'.
        // Subtract from the sender
        balanceOf[_from] -= _value;    // Beosin (Chengdu LianAn) // Alter the token balance of '_from'.
        // Add the same to the recipient
        balanceOf[_to] += _value;    // Beosin (Chengdu LianAn) // Alter the token balance of '_to'.
        emit Transfer(_from, _to, _value);    // Beosin (Chengdu LianAn) // Trigger the event 'Transfer'.
        // Asserts are used to use static analysis to find bugs in your code. They should never fail
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);    // Beosin (Chengdu LianAn) //
Assert that the token balance sum of '_from' and '_to' is the same as the sum before this transfer.
    }
```

```
        /* Transfer tokens
         *
         * Send _value tokens to _to from your account
         *
         * @param _to The address of the recipient
         * @param _value the amount to send
         */
        function transfer(address _to, uint256 _value) public {
            _transfer(msg.sender, _to, _value);    // Beosin (Chengdu LianAn) // Call the internal function
'_transfer' to transfer tokens.
        }



        /* Transfer tokens from other address
         *
         * Send `_value` tokens to `_to` on behalf of `_from`
         *
         * @param _from The address of the sender
         * @param _to The address of the recipient
         * @param _value the amount to send
         */
        function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
            require(_value <= allowance[_from][msg.sender]); // Check allowance    // Beosin (Chengdu LianAn) //
Allowance check, require that the transfer value cannot exceed the allowance between '_from' and
'msg.sender'.
            allowance[_from][msg.sender] -= _value;    // Beosin (Chengdu LianAn) // Update the allowance
between two addresses.
            _transfer(_from, _to, _value);    // Beosin (Chengdu LianAn) // Call the internal function '_transfer'
to transfer tokens.
            return true;
        }



        /* Set allowance for other address
         *
         * Allows _spender to spend no more than _value tokens on your behalf
         *
         * @param _spender The address authorized to spend
         * @param _value the max amount they can spend
         */
        // Beosin (Chengdu LianAn) // Beware that changing an allowance with this method brings the risk that
someone may use both the old and the new allowance by unfortunate transaction ordering. It is
recommended that users reset the allowance to zero, and then set a new allowance.
        // Beosin (Chengdu LianAn) // It is recommended to declare and trigger Approval event in the contract.
        function approve(address _spender, uint256 _value) public
            returns (bool success) {
            allowance[msg.sender][_spender] = _value;    // Beosin (Chengdu LianAn) // Setting the approval
value as '_value'.
```

```
            return true;
        }

            /* Set allowance for other address and notify
         *
         * Allows `_spender` to spend no more than `_value` tokens on your behalf, and then ping the contract about
it
         *
         * @param _spender The address authorized to spend
         * @param _value the max amount they can spend
         * @param _extraData some extra information to send to the approved contract
         */
        function approveAndCall(address _spender, uint256 _value, bytes _extraData)
            public
            returns (bool success) {
            tokenRecipient spender = tokenRecipient(_spender);    // Beosin (Chengdu LianAn) // Get the instance
of the external contract 'spender'.
            if (approve(_spender, _value)) {
                spender.receiveApproval(msg.sender, _value, this, _extraData);    // Beosin (Chengdu LianAn) // If
successfully approved to '_spender', call the function 'receiveApproval' of external contract 'spender' to
response the approval.
                return true;
            }
        }


         /* Destroy tokens
         *
         * Remove _value tokens from the system irreversibly
         *
         * @param _value the amount of money to burn
         */
        function burn(uint256 _value) public returns (bool success) {
            require(balanceOf[msg.sender] >= _value);     // Check if the sender has enough    // Beosin (Chengdu
LianAn) // Require that the token balance of caller is sufficient to destroy.
            balanceOf[msg.sender] -= _value;                  // Subtract from the sender    // Beosin (Chengdu
LianAn) // Update the token balance of caller.
            totalSupply -= _value;                            // Updates totalSupply    // Beosin (Chengdu LianAn)
// Update the total token supply.
            emit Burn(msg.sender, _value);    // Beosin (Chengdu LianAn) // Trigger the event 'Burn'.
            return true;
        }

        /**
         * Destroy tokens from other account
         *
         * Remove _value tokens from the system irreversibly on behalf of _from.
         *
         * @param _from the address of the sender
         * @param _value the amount of money to burn
         */
        function burnFrom(address _from, uint256 _value) public returns (bool success) {
            require(balanceOf[_from] >= _value);                // Check if the targeted balance is enough    //
Beosin (Chengdu LianAn) // Require that the token balance of '_from' is sufficient to destroy.
            require(_value <= allowance[_from][msg.sender]);    // Check allowance    // Beosin (Chengdu
LianAn) // Allowance check, require that the destruction value cannot exceed the allowance between '_from'
and 'msg.sender'.
            balanceOf[_from] -= _value;                         // Subtract from the targeted balance    //
Beosin (Chengdu LianAn) // Alter the token balance of '_from'.
            allowance[_from][msg.sender] -= _value;             // Subtract from the sender's allowance    //
Beosin (Chengdu LianAn) // Update the allowance between two addresses.
            totalSupply -= _value;                              // Update totalSupply    // Beosin (Chengdu
LianAn) // Update the total token supply.
            emit Burn(_from, _value);    // Beosin (Chengdu LianAn) // Trigger the event 'Burn'.
            return true;
        }
}
// Beosin (Chengdu LianAn) // Recommend the main contract to inherit 'Pausable' module to grant owner
the authority of pausing all transactions when serious issue occurred.
```

BEOSIN

Blockchain Security

**Official Website**

https://lianantech.com

**E-mail**

vaas@lianantech.com

**Twitter**

https://twitter.com/Beosin_com