SOLIDITY.*FINANCE*

AlgoVest Token - Audit Report

SUMMARY

AlgoVest.fi is building an advanced AI algorithm trading system for forex and cryptocurrency, which powers their community treasury fund that is used to grow the AlgoVest ecosystem and to provide upward price pressure for the AVS token.

For this audit, we analyzed Algovest's token smart contract, deployed at 0x94d916873b22c9c1b53695f1c002f78537b9b3b2.

> *Features of the token contract:*

- *A burn function exists, allowing any user to burn their own tokens.*

- *Only the tokenRecover function is protected and can only be called by the contract owner. This function allows the owner to retrieve tokens mistakenly sent to the token contract.*
- *The owner can transfer ownership to any address.*

- *The contract includes the ServicePayer and ServiceReceiver libraries - This has no impact on user functionality, however.*
- *Utilization of SafeMath to prevent overflows.*

Audit Findings Summary

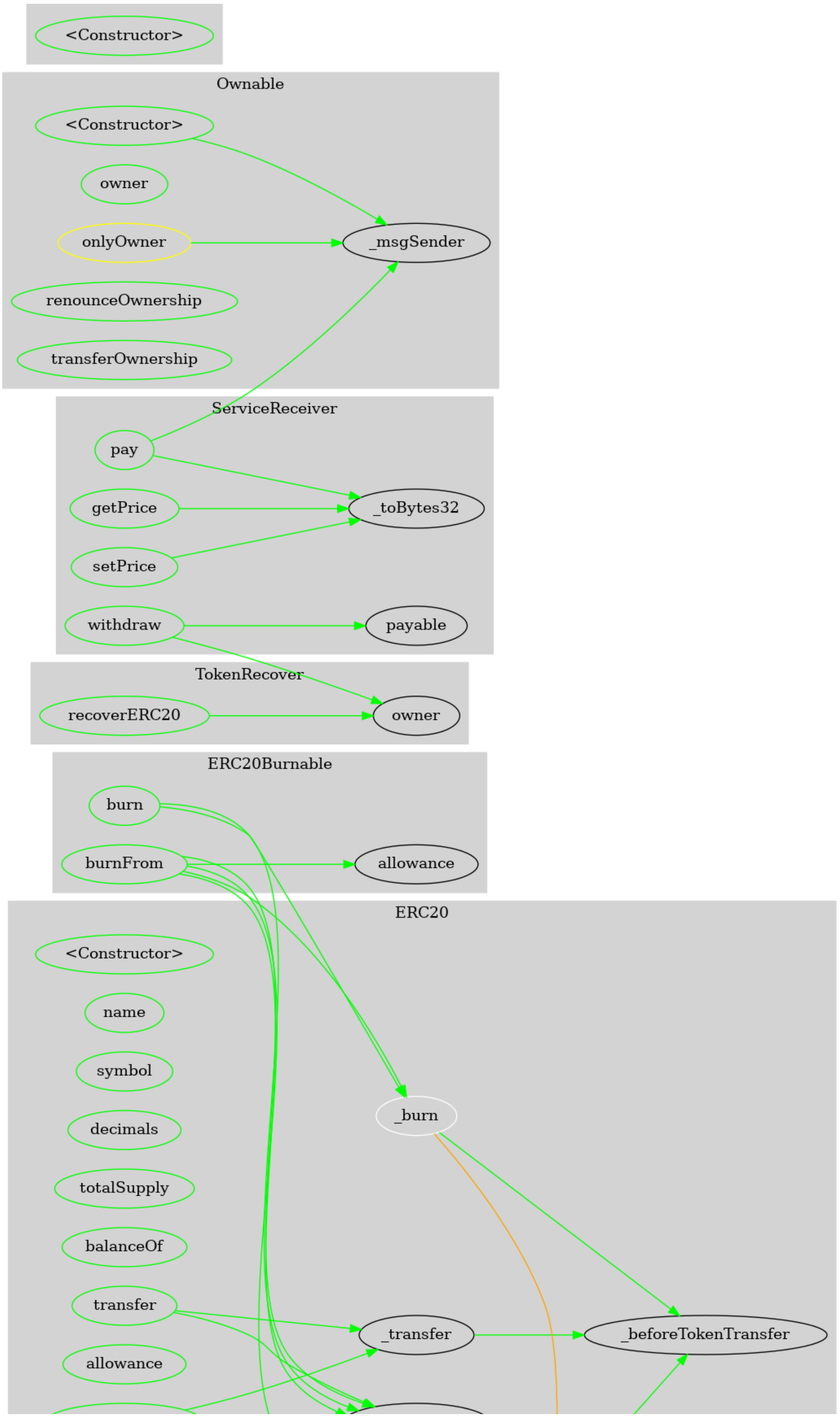
- *No security issues were identified.*
- *Date: January 6th, 2020*

AUDIT RESULTS

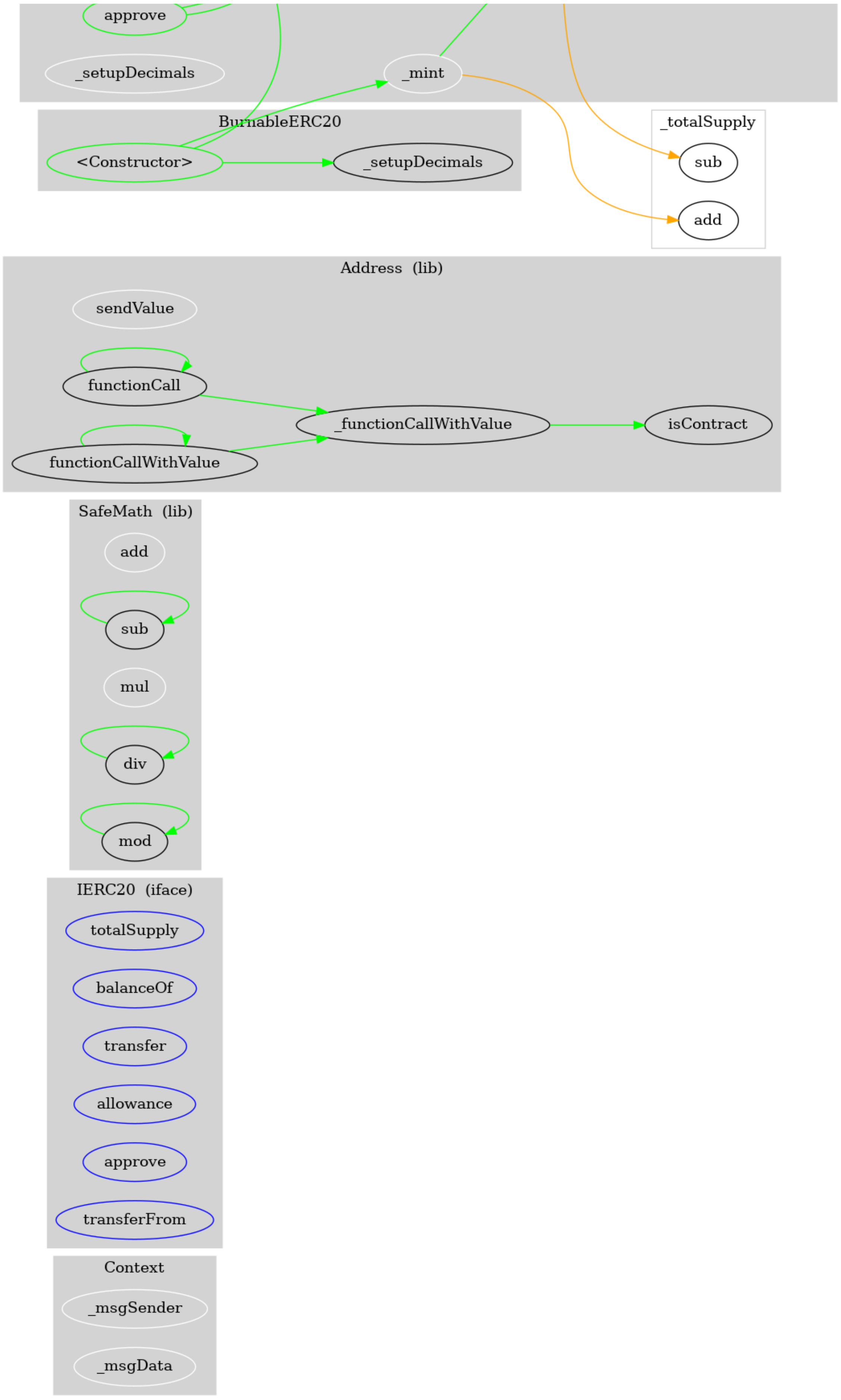
Vulnerability Category	Notes	Result
Arbitrary Storage Write	N/A	PASS
Arbitrary Jump	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS

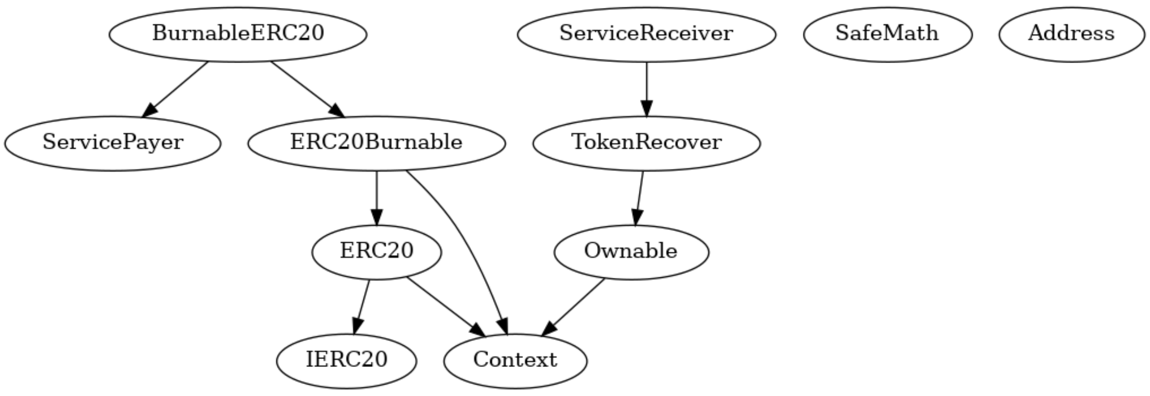
Vulnerability Category	Notes	Result
Deprecated Opcodes	N/A	PASS
Ether Thief	N/A	PASS
Exceptions	N/A	PASS
External Calls	N/A	PASS
Integer Over/Underflow	N/A	PASS
Multiple Sends	N/A	PASS
Suicide	N/A	PASS
State Change External Calls	N/A	Pass
Unchecked Retval	N/A	PASS
User Supplied Assertion	N/A	PASS
Critical Solidity Compiler	N/A	PASS
Overall Contract Safety		PASS

FUNCTION GRAPH



<Constructor>
Ownable
<Constructor>
owner
onlyOwner
_msgSender
renounceOwnership
transferOwnership
ServiceReceiver
pay
getPrice
_toBytes32
setPrice
withdraw
payable
TokenRecover
recoverERC20
owner
ERC20Burnable
burn
burnFrom
allowance
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
_burn
_transfer
_beforeTokenTransfer
allowance


approve
_setupDecimals
_mint
BurnableERC20
<Constructor>
_setupDecimals
_totalSupply
sub
add
Address (lib)
sendValue
functionCall
_functionCallWithValue
isContract
functionCallWithValue
SafeMath (lib)
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData



FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

 +  Context
   - [Int] _msgSender
   - [Int] _msgData

+ [Int] IERC20
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ [Lib] SafeMath
```

```
    - [Int] mul
    - [Int] div
    - [Int] div
    - [Int] mod
    - [Int] mod

 + [Lib] Address
    - [Int] isContract
    - [Int] sendValue #
    - [Int] functionCall #
    - [Int] functionCall #
    - [Int] functionCallWithValue #
    - [Int] functionCallWithValue #
    - [Prv] _functionCallWithValue #

 +  ERC20 (Context, IERC20)
    - [Pub]  #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals
    - [Pub] totalSupply
    - [Pub] balanceOf
    - [Pub] transfer #
    - [Pub] allowance
    - [Pub] approve #
    - [Pub] transferFrom #
    - [Pub] increaseAllowance #
    - [Pub] decreaseAllowance #
    - [Int] _transfer #
    - [Int] _mint #
    - [Int] _burn #
```

```
+  ERC20Burnable (Context, ERC20)
   - [Pub] burn #
   - [Pub] burnFrom #

+  Ownable (Context)
   - [Pub]  #
   - [Pub] owner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner

+  TokenRecover (Ownable)
   - [Pub] recoverERC20 #
      - modifiers: onlyOwner

+  ServiceReceiver (TokenRecover)
   - [Pub] pay ($)
   - [Pub] getPrice
   - [Pub] setPrice #
      - modifiers: onlyOwner
   - [Pub] withdraw #
      - modifiers: onlyOwner
   - [Prv] _toBytes32

+  ServicePayer
   - [Pub]  ($)

+  BurnableERC20 (ERC20Burnable, ServicePayer)
   - [Pub]  ($)
```

SOURCE CODE

Click here to download the source code as a .sol file.

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

// File: @openzeppelin/contracts/GSN/Context.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.7.0;

/*
 * @dev Provides information about the current execu
 * sender of the transaction and its data. While the
 * via msg.sender and msg.data, they should not be a
 * manner, since when dealing with GSN meta-transact
 * paying for execution may not be the actual sender
 * is concerned).
 *
 * This contract is only required for intermediate,
 */
abstract contract Context {
    function _msgSender() internal view virtual retu
        return msg.sender;
    }
    }
}

// File: @openzeppelin/contracts/token/ERC20/IERC20.



pragma solidity ^0.7.0;

/**
 * @dev Interface of the ERC20 standard as defined i
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existenc
     */
    function totalSupply() external view returns (ui

    /**
     * @dev Returns the amount of tokens owned by `a
     */
    function balanceOf(address account) external vie

    /**
     * @dev Moves `amount` tokens from the caller's
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amo
     * allowed to spend on behalf of `owner` through
     * zero by default.
     *
     * This value changes when {approve} or {transfe
     */
    function allowance(address owner, address spende

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * Returns a boolean value indicating whether th
     *
     * IMPORTANT: Beware that changing an allowance 
     * that someone may use both the old and the new
     * transaction ordering. One possible solution t
     * condition is to first reduce the spender's al
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#is
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount

    /**
     * @dev Moves `amount` tokens from `sender` to `
     * allowance mechanism. `amount` is then deducte
     * allowance.
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
    /**
     * @dev Emitted when `value` tokens are moved fr
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address ind

    /**
     * @dev Emitted when the allowance of a `spender
     * a call to {approve}. `value` is the new allow
     */
    event Approval(address indexed owner, address in
}

// File: @openzeppelin/contracts/math/SafeMath.sol



pragma solidity ^0.7.0;

/**
 * @dev Wrappers over Solidity's arithmetic operatio
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflo
 * in bugs, because programmers usually assume that 
 * error, which is the standard behavior in high lev
 * `SafeMath` restores this intuition by reverting t
 * operation overflows.
 *
```

```
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned int
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure
        return sub(a, b, "SafeMath: subtraction over
    }
```

```
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b, string memory
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsign
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure
        // Gas optimization: this is cheaper than re
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/open
        if (a == 0) {
            return 0;
        require(c / a == b, "SafeMath: multiplicatio

        return c;
    }

    /**
     * @dev Returns the integer division of two unsi
     * division by zero. The result is rounded towar
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure
        return div(a, b, "SafeMath: division by zero
    }

    /**
     * @dev Returns the integer division of two unsi
     * division by zero. The result is rounded towar
     *
     * Counterpart to Solidity's `/` operator. Note:
     * `revert` opcode (which leaves remaining gas u
     * uses an invalid opcode to revert (consuming a
     *
     * Requirements:
     *
```

```
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is n

        return c;
    }

    /**
     * @dev Returns the remainder of dividing two un
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This 
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure
        return mod(a, b, "SafeMath: modulo by zero")
    }

    /**
     * @dev Returns the remainder of dividing two un
     * Reverts with custom message when dividing by 
     *
     * Counterpart to Solidity's `%` operator. This 
     * opcode (which leaves remaining gas untouched)
     * invalid opcode to revert (consuming all remai
     *
```

```
         */
        function mod(uint256 a, uint256 b, string memory
            require(b != 0, errorMessage);
            return a % b;
        }
    }

    // File: @openzeppelin/contracts/utils/Address.sol




    pragma solidity ^0.7.0;

    /**
     * @dev Collection of functions related to the addre
     */
    library Address {
        /**
         * @dev Returns true if `account` is a contract.
         *
         * [IMPORTANT]
         * ====
         * It is unsafe to assume that an address for wh
         * false is an externally-owned account (EOA) an
         *
         * Among others, `isContract` will return false
         * types of addresses:
         *
         *  - an externally-owned account
         *  - a contract in construction
         *  - an address where a contract will be create
        function isContract(address account) internal vi
            // According to EIP-1052, 0x0 is the value r
            // and 0xc5d2460186f7233c927e7db2dcc703c0e50
            // for accounts without code, i.e. `keccak25
            bytes32 codehash;
            bytes32 accountHash = 0xc5d2460186f7233c927e
            // solhint-disable-next-line no-inline-assem
            assembly { codehash := extcodehash(account)
            return (codehash != accountHash && codehash
        }

        /**
         * @dev Replacement for Solidity's `transfer`: s
         * `recipient`, forwarding all available gas and
         *
         * https://eips.ethereum.org/EIPS/eip-1884[EIP18
         * of certain opcodes, possibly making contracts
         * imposed by `transfer`, making them unable to
         * `transfer`. {sendValue} removes this limitati
         *
         * https://diligence.consensys.net/posts/2019/09
         *
         * IMPORTANT: because control is transferred to
         * taken to not create reentrancy vulnerabilitie
         * {ReentrancyGuard} or the
         * https://solidity.readthedocs.io/en/v0.5.11/se
         */
        function sendValue(address payable recipient, ui
            require(address(this).balance >= amount, "Ad

            // solhint-disable-next-line avoid-low-level
        /**
         * @dev Performs a Solidity function call using
         * plain`call` is an unsafe replacement for a fu
         * function instead.
         *
         * If `target` reverts with a revert reason, it
         * function (like regular Solidity function call
         *
         * Returns the raw returned data. To convert to
         * use https://solidity.readthedocs.io/en/latest
         *
         * Requirements:
         *
         * - `target` must be a contract.
         * - calling `target` with `data` must not rever
         *
         * _Available since v3.1._
         */
        function functionCall(address target, bytes memo
          return functionCall(target, data, "Address: lo
        }

        /**
         * @dev Same as {xref-Address-functionCall-addre
         * `errorMessage` as a fallback revert reason wh
         *
         * _Available since v3.1._
         */
        function functionCall(address target, bytes memo
            return _functionCallWithValue(target, data,
         * @dev Same as {xref-Address-functionCall-addre
         * but also transferring `value` wei to `target`
         *
         * Requirements:
         *
         * - the calling contract must have an ETH balan
         * - the called Solidity function must be `payab
         *
         * _Available since v3.1._
         */
        function functionCallWithValue(address target, b
            return functionCallWithValue(target, data, v
        }

        /**
         * @dev Same as {xref-Address-functionCallWithVa
         * with `errorMessage` as a fallback revert reas
         *
         * _Available since v3.1._
         */
        function functionCallWithValue(address target, b
            require(address(this).balance >= value, "Add
            return _functionCallWithValue(target, data,
        }

        function _functionCallWithValue(address target,
            require(isContract(target), "Address: call t

            // solhint-disable-next-line avoid-low-level
            (bool success, bytes memory returndata) = ta
            if (success) {
                if (returndata.length > 0) {
                    // The easiest way to bubble the rev

                    // solhint-disable-next-line no-inli
                    assembly {
                        let returndata_size := mload(ret
                        revert(add(32, returndata), retu
                    }
                } else {
                    revert(errorMessage);
                }
            }
        }
    }

    // File: @openzeppelin/contracts/token/ERC20/ERC20.s



    pragma solidity ^0.7.0;






    /**
     * @dev Implementation of the {IERC20} interface.
     *
     * This implementation is agnostic to the way tokens
     * that a supply mechanism has to be added in a deri
     * For a generic mechanism see {ERC20PresetMinterPau
```

```
  * to implement supply mechanisms].
  *
  * We have followed general OpenZeppelin guidelines:
  * of returning `false` on failure. This behavior is
  * and does not conflict with the expectations of ER
  *
  * Additionally, an {Approval} event is emitted on c
  * This allows applications to reconstruct the allow
  * by listening to said events. Other implementation
  * these events, as it isn't required by the specifi
  *
  * Finally, the non-standard {decreaseAllowance} and
  * functions have been added to mitigate the well-kn
  * allowances. See {IERC20-approve}.
  */
contract ERC20 is Context, IERC20 {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    /**
     * @dev Sets the values for {name} and {symbol},
     *
     * All three of these values are immutable: they
     * construction.
     */
    constructor (string memory name_, string memory
        _name = name_;
        _symbol = symbol_;
        _decimals = 18;
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() public view returns (string memo
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually
     * name.
     */
    function symbol() public view returns (string me
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to g
     * For example, if `decimals` equals `2`, a bala
     * be displayed to a user as `5,05` (`505 / 10 *
     *
     * Tokens usually opt for a value of 18, imitati
```

```
         * NOTE: This information is only used for _disp
         * no way affects any of the arithmetic of the c
         * {IERC20-balanceOf} and {IERC20-transfer}.
         */
        function decimals() public view returns (uint8)
            return _decimals;
        }

        /**
         * @dev See {IERC20-totalSupply}.
         */
        function totalSupply() public view override retu
            return _totalSupply;
        }

        /**
         * @dev See {IERC20-balanceOf}.
         */
        function balanceOf(address account) public view
            return _balances[account];
        }

        /**
         * @dev See {IERC20-transfer}.
         *
         * Requirements:
         *
         * - `recipient` cannot be the zero address.
         * - the caller must have a balance of at least
         */
        function transfer(address recipient, uint256 amo
        /**
         * @dev See {IERC20-allowance}.
         */
        function allowance(address owner, address spende
            return _allowances[owner][spender];
        }

        /**
         * @dev See {IERC20-approve}.
         *
         * Requirements:
         *
         * - `spender` cannot be the zero address.
         */
        function approve(address spender, uint256 amount
            _approve(_msgSender(), spender, amount);
            return true;
        }

        /**
         * @dev See {IERC20-transferFrom}.
         *
         * Emits an {Approval} event indicating the upda
         * required by the EIP. See the note at the begi
         *
         * Requirements:
         * - `sender` and `recipient` cannot be the zero
         * - `sender` must have a balance of at least `a
         * - the caller must have allowance for ``sender
         * `amount`.
```

```
        _approve(sender, _msgSender(), _allowances[s
        return true;
    }

    /**
     * @dev Atomically increases the allowance grant
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }

    /**
     * @dev Atomically decreases the allowance grant
     *
     * This is an alternative to {approve} that can
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the upda
     *
     * Requirements:
     *
     */
    function decreaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `
     *
     * This is internal function is equivalent to {t
     * e.g. implement automatic token fees, slashing
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `a
     */
    function _transfer(address sender, address recip
        require(sender != address(0), "ERC20: transf
        require(recipient != address(0), "ERC20: tra

        _beforeTokenTransfer(sender, recipient, amou

        _balances[sender] = _balances[sender].sub(am
        _balances[recipient] = _balances[recipient].
        emit Transfer(sender, recipient, amount);
    }
```

```
     * Emits a {Transfer} event with `from` set to t
     *
     * Requirements
     *
     * - `to` cannot be the zero address.
     */
    function _mint(address account, uint256 amount)
        require(account != address(0), "ERC20: mint

        _beforeTokenTransfer(address(0), account, am

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(
        emit Transfer(address(0), account, amount);
    }

    /**
     * @dev Destroys `amount` tokens from `account`,
     * total supply.
     *
     * Emits a {Transfer} event with `to` set to the
     *
     * Requirements
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` token
     */
    function _burn(address account, uint256 amount)
        require(account != address(0), "ERC20: burn

        _beforeTokenTransfer(account, address(0), am
        emit Transfer(account, address(0), amount);
    }

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This internal function is equivalent to `appr
     * e.g. set automatic allowances for certain sub
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(address owner, address spender
        require(owner != address(0), "ERC20: approve
        require(spender != address(0), "ERC20: appro

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

    /**
     * @dev Sets {decimals} to a value other than th
     *
     * WARNING: This function should only be called 
     * applications that interact with token contrac
     * {decimals} to ever change, and may work incor
     */
    /**
     * @dev Hook that is called before any transfer
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `am
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will b
     * - when `to` is zero, `amount` of ``from``'s t
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:
     */
    function _beforeTokenTransfer(address from, addr
}

// File: @openzeppelin/contracts/token/ERC20/ERC20Bu



pragma solidity ^0.7.0;




/**
 * @dev Extension of {ERC20} that allows token holde
 * tokens and those that they have an allowance for,
 * recognized off-chain (via event analysis).
 */
```

```
    /**
     * @dev Destroys `amount` tokens from the caller
     *
     * See {ERC20-_burn}.
     */
    function burn(uint256 amount) public virtual {
        _burn(_msgSender(), amount);
    }

    /**
     * @dev Destroys `amount` tokens from `account`,
     * allowance.
     *
     * See {ERC20-_burn} and {ERC20-allowance}.
     *
     * Requirements:
     *
     * - the caller must have allowance for ``accoun
     * `amount`.
     */
    function burnFrom(address account, uint256 amoun
        uint256 decreasedAllowance = allowance(accou

        _approve(account, _msgSender(), decreasedAll
        _burn(account, amount);
    }
}

// File: @openzeppelin/contracts/access/Ownable.sol
```

```
/**
 * @dev Contract module which provides a basic acces
 * there is an account (an owner) that can be grante
 * specific functions.
 *
 * By default, the owner account will be the one tha
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will
 * `onlyOwner`, which can be applied to your functio
 * the owner.
 */
abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previ

    /**
     * @dev Initializes the contract setting the dep
     */
    constructor () {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSen
    }

    /**
     * @dev Returns the address of the current owner
     */
    function owner() public view returns (address) {
        return _owner;
     * @dev Throws if called by any account other th
     */
    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: ca
        _;
    }

    /**
     * @dev Leaves the contract without owner. It wi
     * `onlyOwner` functions anymore. Can only be ca
     *
     * NOTE: Renouncing ownership will leave the con
     * thereby removing any functionality that is on
     */
    function renounceOwnership() public virtual only
        emit OwnershipTransferred(_owner, address(0)
        _owner = address(0);
    }

    /**
     * @dev Transfers ownership of the contract to a
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) pub
        require(newOwner != address(0), "Ownable: ne
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

// File: eth-token-recover/contracts/TokenRecover.so
```

```
pragma solidity ^0.7.0;



/**
 * @title TokenRecover
 * @dev Allow to recover any ERC20 sent into the con
 */
contract TokenRecover is Ownable {

    /**
     * @dev Remember that only owner can call so be 
     * @param tokenAddress The token contract addres
     * @param tokenAmount Number of tokens to be sen
     */
    function recoverERC20(address tokenAddress, uint
        IERC20(tokenAddress).transfer(owner(), token
    }
}

// File: contracts/service/ServiceReceiver.sol



pragma solidity ^0.7.0;



/**
 * @title ServiceReceiver
 * @dev Implementation of the ServiceReceiver
 */
```

```
    event Created(string serviceName, address indexe

    function pay(string memory serviceName) public p
        require(msg.value == _prices[_toBytes32(serv

        emit Created(serviceName, _msgSender());
    }

    function getPrice(string memory serviceName) pub
        return _prices[_toBytes32(serviceName)];
    }

    function setPrice(string memory serviceName, uin
        _prices[_toBytes32(serviceName)] = amount;
    }

    function withdraw(uint256 amount) public onlyOwn
        payable(owner()).transfer(amount);
    }

    function _toBytes32(string memory serviceName) p
        return keccak256(abi.encode(serviceName));
    }
}

// File: contracts/service/ServicePayer.sol



pragma solidity ^0.7.0;
```

```
 * @title ServicePayer
 * @dev Implementation of the ServicePayer
 */
abstract contract ServicePayer {

    constructor (address payable receiver, string me
        ServiceReceiver(receiver).pay{value: msg.val
    }
}

// File: contracts/token/ERC20/BurnableERC20.sol



pragma solidity ^0.7.0;



/**
 * @title BurnableERC20
 * @dev Implementation of the BurnableERC20
```

PRINT EXPANDED SECTIONS

GO HOME