SOLIDITY.*FINANCE*

Bird.Money Token - Audit Report

SUMMARY

Bird.Money ($BIRD) intends to build an off-chain oracle analytics and lending platform. Their oracle and lending applications are still in development, but they have a live API that analyzes and provides a 0-10 assessment score on provided Ethereum addresses. For this audit, we analyzed the project team's native token contract, deployed at 0x70401dFD142A16dC7031c56E862Fc88Cb9537Ce0.

> *Notes of the token contract:*
>
> - *No accessible mint functions exist. After deployment, the total*

- *The team has burned tokens since deployment, and the total supply is now 140,000 tokens; about 50% of which are owned by the team. Over 90% of the team's tokens were locked on November 25th for 30, 60, & 90 days; and 5 years (Approximate amount of BIRD tokens, respectively: 4.5k, 27.3k, 4.5k, 13.6k).*
- *$250,000 of liquidity (26.7k BIRD + 205 ETH) has been locked in Unicrypt for 6 months. More details on the updated tokenomics are here.*
- *No owner-restricted functions are present.*
- *Utilization of SafeMath to prevent overflows.*

Audit Findings:

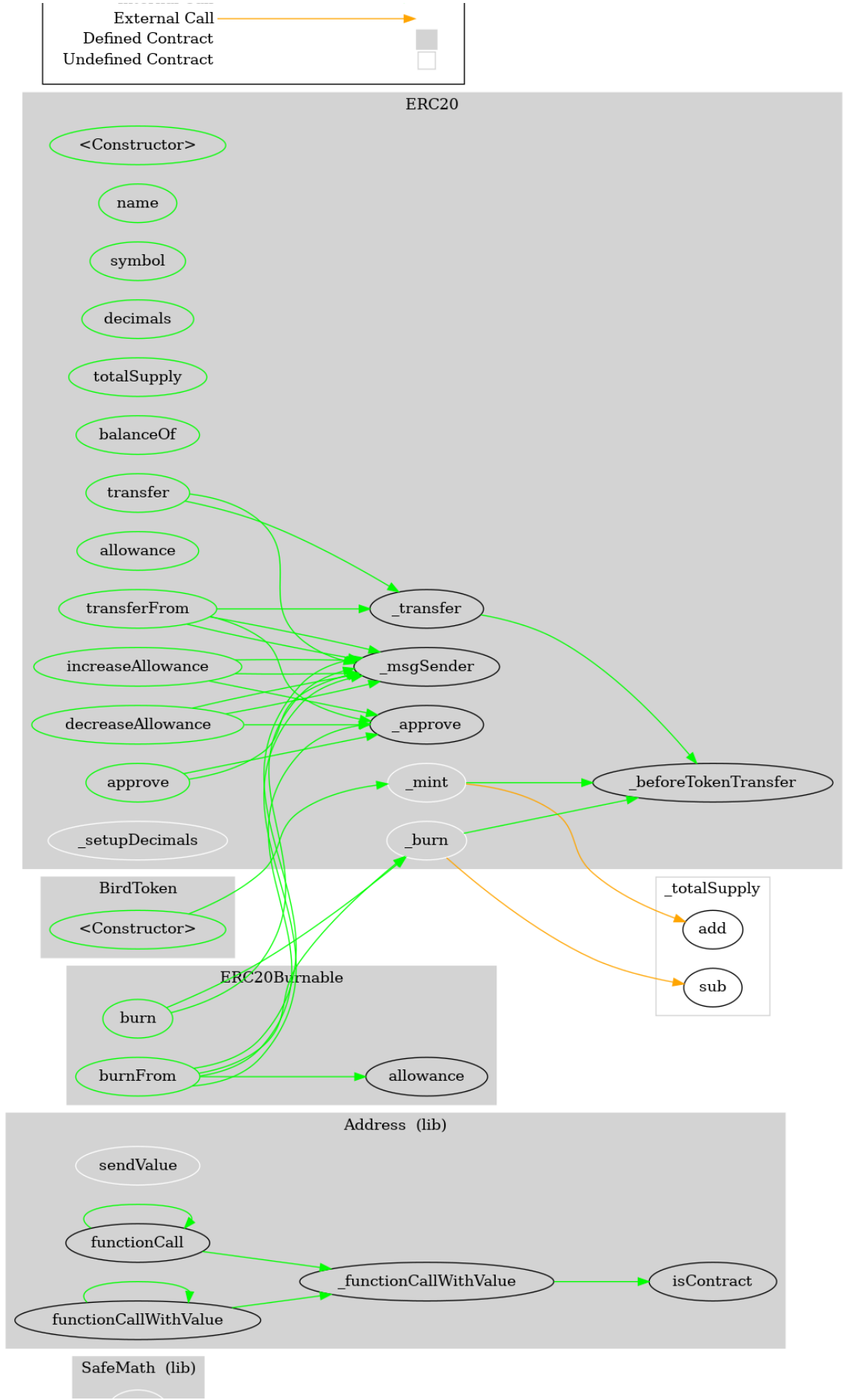
- *No security vulnerabilities from external or internal actors were identified.*
- *Date: December 4th, 2020*

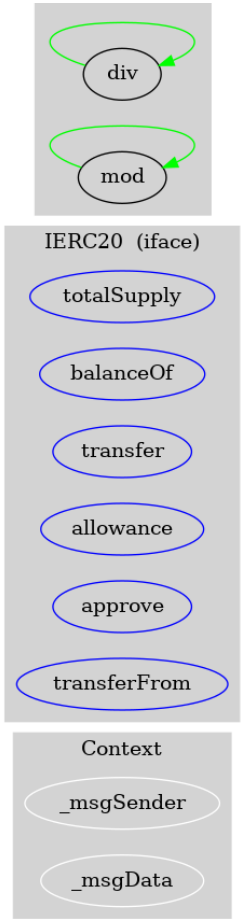
AUDIT RESULTS

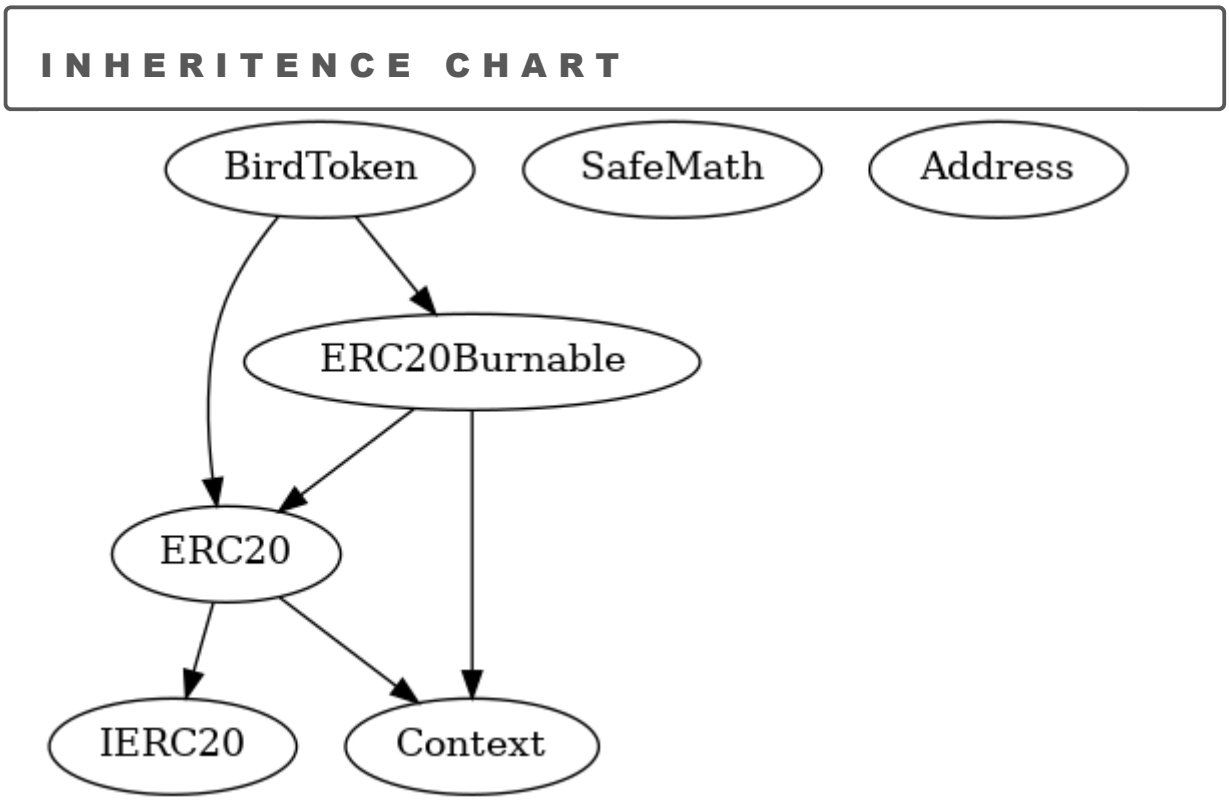
Date: December 4th, 2020

Vulnerability Category	Notes	Result
Arbitrary Storage Write	N/A	PASS

Vulnerability Category	Notes	Result
Dependence on Predictable Variables	N/A	PASS
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







```
($) = 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] add
   - [Int] sub
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] div
   - [Int] mod
   - [Int] mod

+ [Lib] Address
```

```
    - [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 #
    - [Int] _approve #
    - [Int] _setupDecimals #
    - [Int] _beforeTokenTransfer #

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

+   BirdToken (ERC20, ERC20Burnable)
    - [Pub]  #
```

SOURCE CODE

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

```
/*
      $$$$$$$\  $$\                  $$\      $$\
      $$  __$$\ \__|                 $$ |     $$$\
      $$ |  $$ |$$\  $$$$$$\   $$$$$$$ |     $$$$\  $
      $$$$$$$\ |$$ |$$  __$$\ $$  __$$ |     $$\$$\$$
      $$  __$$\ $$ |$$ |  \__|$$ /  $$ |     $$ \$$$
      $$ |  $$ |$$ |$$ |      $$ |  $$ |     $$ |\$
      $$$$$$$  |$$ |$$ |      \$$$$$$$ |$$\ $$ | \_/
      \_______/ \__|\__|       \_______|\__|\__|


*/


// Bird.Money Token $BIRD
// © 2020 Bird Money
// SPDX-License-Identifier: MIT

pragma solidity 0.6.12;

/*
 * @dev Provides information about the current execu
```

```
 * 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;
    }

    function _msgData() internal view virtual return
        this; // silence state mutability warning wi
        return msg.data;
    }
}



/**
 * @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
```

```
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amo

    /**
     * @dev Returns the remaining number of tokens t
     * 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
```

```
     * allowance mechanism. `amount` is then deducte
     * allowance.
     *
     * Returns a boolean value indicating whether th
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address re

    /**
     * @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
}

/**
 * @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
```

```
 * Using this library instead of the unchecked opera
 * class of bugs, so it's recommended to use it alwa
 */
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.
     */
```

```
    /**
     * @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, 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.
```

```
        }

        uint256 c = a * b;
        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
```

```
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory
        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
```

```
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory
        require(b != 0, errorMessage);
        return a % b;
    }
}

/**
 * @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
     *  - an address where a contract lived, but was
     * ====
     */
```

```
        // 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
        (bool success, ) = recipient.call{ value: am
        require(success, "Address: unable to send va
    }
```

```
     * 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,
    }

    /**
```

```
     * 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) {
            return returndata;
        } else {
            // Look for revert reason and bubble it
```

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

/**
 * @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
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-impleme
 * 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
```

```
 * 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},
     * a default value of 18.
     *
     * To select a different value for {decimals}, u
     *
     * All three of these values are immutable: they
     * construction.
     */
    constructor (string memory name, string memory s
        _name = name;
        _symbol = symbol;
        _decimals = 18;
    }
```

```
     */
    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
     * Ether and Wei. This is the value {ERC20} uses
     * called.
     *
     * 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;
    }

    /**
```

```
        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
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spende
        return _allowances[owner][spender];
    }

    /**
```

```
     *
     * - `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`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, 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}.
```

```
     * 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:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the calle
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint
        _approve(_msgSender(), spender, _allowances[
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `
     *
```

```
     * 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);
    }

    /** @dev Creates `amount` tokens and assigns the
     * the total supply.
     *
     * 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
```

```
        _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

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

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This is internal function is equivalent to `a
     * e.g. set automatic allowances for certain sub
     *
```

```
     *
     * - `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
     */
    function _setupDecimals(uint8 decimals_) interna
        _decimals = decimals_;
    }

    /**
     * @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`.
```

```
     *
     * To learn more about hooks, head to xref:ROOT:
     */
    function _beforeTokenTransfer(address from, addr
}

/**
 * @dev Extension of {ERC20} that allows token holde
 * tokens and those that they have an allowance for,
 * recognized off-chain (via event analysis).
 */
abstract contract ERC20Burnable is Context, ERC20 {
    /**
     * @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`.
```

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

PRINT EXPANDED SECTIONS

GO HOME