10 MAY 2021

Bifi SMART CONTRACT AUDIT REPORT

Hexlant.

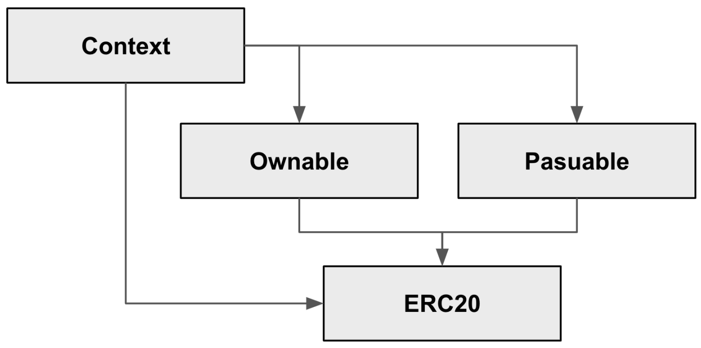
Analysis Purpose

본 리포트는 발행된 컨트렉트 코드가 요구사항을 충분히 만족하는지, 그리고 보안의 취약점과 실제 운영 하면서 발생 할 수 있는 문제들을 파악하고 해결방안을 찾기위해 분석을 수행하고 그 결과를 정리하였습니다. 이번 코드 분석은 다음과 같은 요소들을 검증하기위해 진행하였습니다.

- 구현된 기능의 정상작동 여부
- 기능 수행 중 보안 위험성
- **Off Chain**에서 발생하는 문제에 대한 대비
- 컨트렉트 코드의 가독성 및 코드 완성도

Fuction Summary

Bifi 컨트랙트는 **Open-Zeppelin**에서 제공하는 컨트랙트 코드와 직접 구현한 컨트랙트 코드로 작성되었으며**,** 다음과 같은 컨트랙트를 통해 **Bifi**의 기능을 구현하였습니다**.**



- **Context**
  현재 실행 컨텍스트에 대한 정보를 제공합니다**.**

- **Ownable**
  Ownership과 관련된 기능을 제공합니다**. onlyOwner modifier**을 사용하여 컨트랙트의 **Owner**일 경우 실행되도록 기능을 제공할 수 있습니다**.**

- **Pausable**
  컨트랙트 내 토큰 전송 정지와 관련된 기능을 제공합니다**. whenNotPaused, whenPaused modifier**를 사용하여 토큰 전송 시 현재 컨트랙트의 유통 제제 상황을 확인 후 전송이 이루워지도록 합니다**.**

- **ERC20**
  Bifi의 기본 정보를 정의하고 **ERC20**에 대한 세부 기능을 구현합니다**.**

Contract

상태 변수와 함수를 포함하여
컨테이너 형태의 계약을 표현하기 위해
사용

Contract	Description
Context	컨트랙트 실행 컨텍스트 정보 관리
Ownable	컨트랙트 오너 권한 관리
Pausable	컨트랙트 토큰 정지 상태 관리
ERC20	메인 컨트랙트

Interface

컨트랙트 내 구현하고자 하는
표준함수를 정의하기 위해 사용

Interface	Description
IERC20	ERC20 표준 인터페이스

Library

상태 변수를 갖을 수 없고 상속을
지원하지 않는 컨트랙트 라이브러리로,
라이브러리 내 함수가 호출되며
호출한 컨트랙트의 컨텍스트에서 실행

Library	Description
SafeMath	산술 연산시 발생가능 한 이슈 제어
Address	주소 유형과 관련된 기능

Variable

컨트랙트의 상태를 표현하는
변수들로, 컨트랙트에 필요한
정보들을 저장하기위해 사용

Variable	Description
_paused	컨트랙트 토큰 전송 정지 상태
_owner	컨트랙트 오너 주소
_balances	특정 주소의 토큰 잔액 해시테이블
_allowances	특정 주소에게 출금 위임된 토큰 잔액 해시 테이블
_totalSupply	토큰 총 발행량
_name	토큰의 이름
_symbol	토큰의 심볼
_decimals	토큰의 최대 표현 가능한 소수점 자리수

Modifier

함수의 한정요소로, 특정 기능을
수행할 때 한정된 조건에서만
실행 될 수 있도록 하기 위해

Modifier	Description
whenNotPaused	컨트랙트 토큰 전송 정지 상태가 아닐 경우 실행 가능
whenPaused	컨트랙트 토큰 전송 정지 상태일 경우 실행 가능
onlyOwner	컨트랙트 오너일 경우 실행 가능

Event

컨트랙트 함수 실행에 따른 로그 이벤트로 추후 애플리케이션 적용에 있어 컨트랙트 상황을 보다 쉽게 대응하기 위해 사용

Event	Description
Paused	컨트랙트 토큰 전송 상태로 전환 시 이벤트 발생
UnPaused	컨트랙트 토큰 전송 상태가 해제 시 이벤트 발생
OwnershipTransferred	컨트랙트 오너 권한 이전 시 이벤트 발생
Transfer	토큰 전송 시 이벤트 발생
Approval	토큰 출금 위임 시 이벤트 발생

Function

컨트랙트의 함수들 로서 컨트랙트에
필요한 특정 로직을 담아
기능 실행을 하기 위해 사용

Function	Description
_msgSender	기능 호출 주체 (발신자)
_msgData	calldata
paused	컨트랙트 토큰 전송 정시 상태 확인
_pause	컨트랙트 토큰 전송 정지
_unpause	컨트랙트 토큰 전송 정지 해제
owner	컨트랙트 오너 주소 확인
renounceOwnership	컨트랙트 오너 권한 포기
transferOwnership	컨트랙트 오너 권한 이전
name	토큰의 이름 확인
symbol	토큰의 심볼 확인
decimals	토큰의 데시멀 확인
totalSupply	토큰의 총 발행량 확인
balanceOf	특정 주소의 토큰 잔액 확인
transfer	특정 주소에게 토큰 전송
allowance	특정 주소에게 출금 위임된 토큰 잔액 확인
approve	특정 주소에게 출금 위임
transferFrom	특정 주소에게 출금 위임된 토큰 전송
increaseAllowance	특정 주소에게 출금 위임된 토큰 잔액 증액
decreaseAllowance	특정 주소에게 출금 위임된 토큰 잔액 감액
_transfer	특정 주소에게 토큰 전송
_burn	토큰 소각
_approve	특정 주소에게 출금 위임
burn	토큰 소각
burnFrom	출금 위임된 토큰 소각
pause	컨트랙트 토큰 전송 정지
unpause	컨트랙트 토큰 전송 정지 해제
_beforeTokenTransfer	토큰 전송 전 컨트랙트 토큰 전송 정지 상태 확인

Fuction Profile

Function Profile에서는 컨트랙트 함수들의 세부적인 내용들을 표시합니다. 함수의 세부적인 매개변수와, 다양한 옵션을 살펴보고, 콜스택을 통해 함수간의 호출관계를 정리합니다. 이를 통해 함수 호출관계를 파악하고, 함수들 간의 논리적 충돌이 있는지 쉽게 파악 할 수 있습니다.

Function Name	(Context) _msgSender		
Parameter	-	**Return**	address
Visibility	internal	**Modifier**	-
Constant	view	**Inheritance**	virtual
Callstack			
_msgSender			

Function Name	(Context) _msgData		
Parameter	-	**Return**	bytes
Visibility	internal	**Modifier**	-
Constant	view	**Inheritance**	virtual
Callstack			
_msgData			

Function Name	(Pausable) paused		
Parameter	-	**Return**	bool
Visibility	public	**Modifier**	-
Constant	view	**Inheritance**	-
Callstack			
paused			

Function Name	(Pausable) _pause		
Parameter	-	**Return**	-
Visibility	internal	**Modifier**	whenNotPaused
Constant	-	**Inheritance**	virtual
Callstack			
_pause			

Function Name	(Pausable) _unpause		
Parameter	-	**Return**	-
Visibility	internal	**Modifier**	whenPaused

Constant	-	**Inheritance**	virtual
Callstack			
_unpause			

Function Name	(Ownable) owner		
Parameter	-	**Return**	-
Visibility	public	**Modifier**	-
Constant	view	**Inheritance**	-
Callstack			
owner			

Function Name	(Ownable) renounceOwnership		
Parameter	-	**Return**	-
Visibility	public	**Modifier**	onlyOwner
Constant	-	**Inheritance**	virtual
Callstack			
renounceOwnership			

Function Name	(Ownable) transferOwnership		
Parameter	-	**Return**	-
Visibility	public	**Modifier**	onlyOwner
Constant	-	**Inheritance**	virtual
Callstack			
transferOwnership			

Function Name	(ERC20) name		
Parameter	-	**Return**	string
Visibility	public	**Modifier**	-
Constant	view	**Inheritance**	-
Callstack			
name			

Function Name	(ERC20) symbol		
Parameter	-	Return	string
Visibility	public	Modifier	-
Constant	view	Inheritance	-
Callstack			
symbol			

Function Name	(ERC20) decimals		
Parameter	-	Return	uint8
Visibility	public	Modifier	-
Constant	view	Inheritance	-
Callstack			
decimals			

Function Name	(ERC20) totalSupply		
Parameter	-	Return	uint256
Visibility	public	Modifier	-
Constant	view	Inheritance	override
Callstack			
totalSupply			

Function Name	(ERC20) balanceOf		
Parameter	address	Return	uint256
Visibility	public	Modifier	-
Constant	view	Inheritance	overrid
Callstack			
balanceOf			

Function Name	(ERC20) transfer		
Parameter	address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual, override
Callstack			
transfer ⌊ _transfer ⌊ _msgSender			

Function Name	(ERC20) allowance		
Parameter	address, address	Return	uint256
Visibility	public	Modifier	-
Constant	view	Inheritance	virtual, override
Callstack			
allowance			

Function Name	(ERC20) approve		
Parameter	address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual, override
Callstack			
approve ㄴ_approve ㄴ_msgSender			

Function Name	(ERC20) transferFrom		
Parameter	address, address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual, override
Callstack			
transferFrom ㄴ_transfer ㄴ_approve ㄴ_msgSender			

Function Name	(ERC20) increaseAllowance		
Parameter	address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual
Callstack			

increaseAllowance
⎿_approve
⎿ _msgSender

Function Name	(ERC20) decreaseAllowance		
Parameter	address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual
Callstack			

decreaseAllowance
⎿ _approve
⎿ _msgSender

Function Name	(ERC20) _transfer		
Parameter	address, address, uint256	Return	bool
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual
Callstack			

_transfer
⎿ _beforeTokenTransfer

Function Name	(ERC20) _burn		
Parameter	address, uint256	Return	-
Visibility	internal	Modifier	-
Constant	-	Inheritance	virtual
Callstack			

_burn
⎿ _beforeTokenTransfer

Function Name	(ERC20) _approve		
Parameter	address, address, uint256	Return	-
Visibility	internal	Modifier	-
Constant	-	Inheritance	virtual
Callstack			
_approve			

Function Name	(ERC20) burn		
Parameter	uint256	Return	-
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual
Callstack			
burn ⌊ _burn ⌊ _msgSender			

Function Name	(ERC20) burnFrom		
Parameter	address, uint256	Return	-
Visibility	public	Modifier	-
Constant	-	Inheritance	virtual
Callstack			
burnFrom ⌊ allowance ⌊ _msgSender ⌊ _approve ⌊ _msgSender ⌊ _burn			

Function Name	(ERC20) pause		
Parameter	-	Return	-
Visibility	public	Modifier	onlyOwner
Constant	-	Inheritance	-
Callstack			

pause
⎣ _pause

Function Name	(ERC20) unpause		
Parameter	-	Return	-
Visibility	public	Modifier	onlyOwner
Constant	-	Inheritance	-
Callstack			

unpause
⎣ _unpause

Function Name	(ERC20) _beforeTokenTransfer		
Parameter	address, address, uint256	Return	-
Visibility	internal	Modifier	-
Constant	-	Inheritance	virtual
Callstack			

_beforeTokenTransfer
⎣ _paused

Test Result

Code Coverage

코드 커버리지는 작성한 테스트가 얼마만큼 컨트랙트 코드의 기능들을 테스트 했는지 알 수있는 정량적인 지표입니다.

Bifi 컨트랙트는 라이브러리와 일부 컨트랙트에 구현된 기능에 대해 추가적인 호출이 존재하지 않습니다.

아래의 Coverage 지표는 위 사항을 반영한 결과입니다.

File Name	Statements	Functions	Lines
ERC20.sol	100% (67/67)	100% (36/36)	100% (70/70)

Test cases

Test case	Result
토큰 이름은 지정한 이름과 일치한다	PASS
토큰 심볼은 지정한 심볼과 일치한다.	PASS
토큰 데시멀은 지정한 데시멀과 일치한다.	PASS
지정한 초기 발행량이 총 발행량으로 할당된다.	PASS
지정한 초기 발행량이 컨트랙트의 오너(배포 실행 주소)에게 할당된다.	PASS
배포 후 오너 외 주소들의 토큰 잔액은 0이다.	PASS
토큰 전송 시 받는 주소가 0x00 일 경우 예외처리가 되는가?	PASS
토큰 전송 시 보내는 수량이 음수 일 경우 예외처리가 되는가?	PASS
토큰 전송 시 보유한 수량을 초과한 경우 예외처리가 되는가?	PASS
특정 주소에게 출금 위임 시 해당 주소의 출금 위임 잔액이 증액하는가?	PASS
특정 주소에게 출금 위임된 토큰 잔액을 증액 혹은 감액할 수 있는가?	PASS
특정 주소에게 출금 위임된 토큰 잔액 이상을 감액시키면 0이 되는가?	PASS
출금 위임받은 토큰 전송이 가능한가?	PASS
출금 위임받은 토큰 전송 시 관련 주소들의 토큰 잔액이 올바르게 업데이트 되는가?	PASS
출금 위임받은 토큰 전송 시 받는 주소가 0일 경우 예외처리가 되는가?	PASS
출금 위임받은 토큰 잔액을 초과한 수량을 전송 시 예외처리가 되는가?	PASS
출금을 위임한 주소의 토큰 보유 잔액이 부족할 경우 예외처리가 되는가?	PASS
컨트랙트 정지 관리자 외 주소로부터 토큰 전송 정지 기능 실행 시 예외처리가 되는가?	PASS

Test case	Result
컨트랙트 토큰 전송 정지 상태에서 토큰 전송 시 예외처리가 되는가?	PASS
컨트랙트 토큰 전송 정지 상태에서 출금 위임을 통한 토큰 전송 시 예외처리가 되는가?	PASS
컨트랙트 토큰 정지 상태를 해제 가능한가?	PASS
컨트랙트 오너주소를 확인할 수 있는가?	PASS
컨트랙트 오너는 자신의 권한을 포기할 수 있으며 포기 이후 오너 주소는 0x0이 되는가?	PASS
컨트랙트 오너 외의 주소로부터 오너 권한 이전 시 예외처리가 되는가?	PASS
컨트랙트 오너는 자신의 권한 이전할 수 있는가?	PASS
보유 잔액을 초과한 토큰 소각 시 예외처리가 되는가?	PASS
보유 출금 위임 잔액을 초과한 토큰 소각 시 예외처리가 되는가?	PASS
토큰 소각 시 보유 잔액 및 총 발행량이 함께 감액되는가?	PASS
출금 위임된 토큰 소각 시 위임자의 보유 잔액 및 총 발행량이 함께 감액되는가?	PASS
토큰 전송 시 이벤트가 발생하는가?	PASS
특정 주소에게 출금 위임 시 이벤트가 발생하는가?	PASS
특정 주소에게 출금 위임 된 토큰 잔액을 증액 혹은 감액 시 이벤트가 발생하는가?	PASS
특정 주소에게 출금 위 임 된 토큰 전송 시 이벤트가 발생하는가?	PASS
컨트랙트 오너 권한을 이전 시 이벤트가 발생하는가?	PASS
토큰 소각 시 이벤트가 발생하는가?	PASS

```
/**
 *Submitted for verification at Etherscan.io on 2021-01-04
*/

pragma solidity ^0.6.0;

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
contract Context {
    // Empty internal constructor, to prevent people from mistakenly deploying
    // an instance of this contract, which should be used via inheritance.
    constructor () internal { }

    function _msgSender() internal view virtual returns (address payable) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see https://github.com/
ethereum/solidity/issues/2691
        return msg.data;
    }
}

/**
 * @dev Contract module which allows children to implement an emergency stop
 * mechanism that can be triggered by an authorized account.
 *
 * This module is used through inheritance. It will make available the
 * modifiers `whenNotPaused` and `whenPaused`, which can be applied to
 * the functions of your contract. Note that they will not be pausable by
 * simply including this module, only once the modifiers are put in place.
 */
contract Pausable is Context {
    /**
     * @dev Emitted when the pause is triggered by `account`.
     */
    event Paused(address account);

    /**
     * @dev Emitted when the pause is lifted by `account`.
     */
    event Unpaused(address account);

    bool private _paused;

    /**
     * @dev Initializes the contract in unpaused state.
     */
    constructor () internal {
        _paused = false;
    }
    /**
     * @dev Returns true if the contract is paused, and false otherwise.
     */
    function paused() public view returns (bool) {
        return _paused;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     *
     * Requirements:
     *
     * - The contract must not be paused.
     */
    modifier whenNotPaused() {
        require(!_paused, "Pausable: paused");
        _;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is paused.
     *
     * Requirements:
     *
     * - The contract must be paused.
     */
    modifier whenPaused() {
        require(_paused, "Pausable: not paused");
        _;
    }

    /**
     * @dev Triggers stopped state.
     *
     * Requirements:
     *
     * - The contract must not be paused.
     */
    function _pause() internal virtual whenNotPaused {
        _paused = true;
        emit Paused(_msgSender());
    }

    /**
     * @dev Returns to normal state.
     *
     * Requirements:
     *
     * - The contract must be paused.
     */
    function _unpause() internal virtual whenPaused {
        _paused = false;
        emit Unpaused(_msgSender());
    }
}
// SPDX-License-Identifier: MIT
/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    /**
     * @dev Initializes the contract setting the deployer as the initial owner.
     */
    constructor () internal {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSender);
    }

    /**
     * @dev Returns the address of the current owner.
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: caller is not the owner");
        _;
    }

    /**
     * @dev Leaves the contract without owner. It will not be possible to call
     * `onlyOwner` functions anymore. Can only be called by the current owner.
     *
     * NOTE: Renouncing ownership will leave the contract without an owner,
     * thereby removing any functionality that is only available to the owner.
     */
    function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);
    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: 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. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

```
/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

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

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting with custom message on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsigned integers, reverting on
     * overflow.
     *
    * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    /**
     * @dev Returns the integer division of two unsigned integers. Reverts on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    /**
     * @dev Returns the integer division of two unsigned integers. Reverts with custom message on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function uses a
     * `revert` opcode (which leaves remaining gas untouched) while Solidity
     * uses an invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

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

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
     * Reverts with custom message when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}


pragma solidity ^0.6.0;

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // According to EIP-1052, 0x0 is the value returned for not-yet created accounts
        // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is returned
        // for accounts without code, i.e. `keccak256('')`
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
        return (codehash != accountHash && codehash != 0x0);
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-
interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
        (bool success, ) = recipient.call.value(amount)("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }
}



/**
 * @dev Implementation of the {IERC20} interface.
 *
 * This implementation is agnostic to the way tokens are created.
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms/226[How
 * to implement supply mechanisms].
 *
 * We have followed general OpenZeppelin guidelines: functions revert instead
 * of returning `false` on failure. This behavior is nonetheless conventional
 * and does not conflict with the expectations of ERC20 applications.
 *
 * Additionally, an {Approval} event is emitted on calls to {transferFrom}.
 * This allows applications to reconstruct the allowance for all accounts just
 * by listening to said events. Other implementations of the EIP may not emit
 * these events, as it isn't required by the specification.
 *
 * Finally, the non-standard {decreaseAllowance} and {increaseAllowance}
 * functions have been added to mitigate the well-known issues around setting
 * allowances. See {IERC20-approve}.
 */
contract ERC20 is Context, IERC20, Pausable, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _balances;

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

    uint256 private _totalSupply;
    string private constant _name = "Bifi";
    string private constant _symbol = "Bifi";
    uint8 private constant _decimals = 18;

    /**
     * @dev Sets the values for {name} and {symbol}, initializes {decimals} with
     * a default value of 18.
     *
     * To select a different value for {decimals}, use {_setupDecimals}.
     *
     * All three of these values are immutable: they can only be set once during
     * construction.
     */
    constructor () public {
        _totalSupply = (10 ** 9) * (10 ** uint256(_decimals));
        _balances[msg.sender] = _totalSupply;
    }

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

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

    /**
     * @dev Returns the number of decimals used to get its user representation.
     * For example, if `decimals` equals `2`, a balance of `505` tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
     *
     * Tokens usually opt for a value of 18, imitating the relationship between
     * Ether and Wei. This is the value {ERC20} uses, unless {_setupDecimals} is
     * called.
     *
     * NOTE: This information is only used for _display_ purposes: it in
     * no way affects any of the arithmetic of the contract, including
     * {IERC20-balanceOf} and {IERC20-transfer}.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }

    /**
     * @dev See {IERC20-totalSupply}.
     */
    function totalSupply() public view override returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev See {IERC20-balanceOf}.
     */
    function balanceOf(address account) public view override returns (uint256) {
        return _balances[account];
    }

    /**
     * @dev See {IERC20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
    function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }
    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spender) public view virtual override returns (uint256) {
        return _allowances[owner][spender];
    }

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

    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance. This is not
     * required by the EIP. See the note at the beginning of {ERC20};
     *
     * Requirements:
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     * - the caller must have allowance for ``sender``'s tokens of at least
     * `amount`.
     */
    function transferFrom(address sender, address recipient, uint256 amount) public virtual override
returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer
amount exceeds allowance"));
        return true;
    }

    /**
     * @dev Atomically increases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
        return true;
    }

    /**
     * @dev Atomically decreases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    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;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `recipient`.
     *
     * This is internal function is equivalent to {transfer}, and can be used to
     * e.g. implement automatic token fees, slashing mechanisms, etc.
     *
     * 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 `amount`.
     */
    function _transfer(address sender, address recipient, uint256 amount) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }

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

        _beforeTokenTransfer(account, address(0), amount);

        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }

    /**
     * @dev Sets `amount` as the allowance of `spender` over the `owner`s tokens.
     *
     * This is internal function is equivalent to `approve`, and can be used to
     * e.g. set automatic allowances for certain subsystems, etc.
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(address owner, address spender, uint256 amount) internal virtual {
        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);
    }

    /**
     * @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`, deducting from the caller's
     * allowance.
     *
     * See {ERC20-_burn} and {ERC20-allowance}.
     *
     * Requirements:
     *
     * - the caller must have allowance for ``accounts``'s tokens of at least
     * `amount`.
     */
    function burnFrom(address account, uint256 amount) public virtual {
        uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount 
exceeds allowance");

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

    /**
     * @dev Triggers stopped state.
     *
     * Requirements:
     *
     * - The contract must not be paused.
     */

    function pause() public onlyOwner {
        _pause();
    }

    /**
     * @dev Returns to normal state.
     *
     * Requirements:
     *
     * - The contract must be paused.
     */
    function unpause() public onlyOwner {
        _unpause();
    }
    /**
     * @dev Hook that is called before any transfer of tokens. This includes
     * burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
     * will be to transferred to `to`.
     * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:using-hooks.adoc[Using Hooks].
     */
    function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual {
        require(!paused(), "ERC20Pausable: token transfer while paused");
    }
}
```

Vulnerability Analysis

Critical Severity

심각성 치명적 단계는 일반적인 상황에서 보안 또는 큰 문제를 야기 할 수 있는 오류로
반드시 수정해야 하는 항목입니다

해당 항목 없음

High Severity

심각성 높음 단계는 일반적인 상황에서 발생하는 문제는 아니지만, 특수한 조건이나
예외상황에 의해서 문제가 발생 할 수 있는 항목입니다.
추가적인 예외처리나, 코너케이스에 대하여 분석하고 오류를 막을 수 있도록 수정이 필요한 항목입니다.

해당 항목 없음

Medium Severity

심각성 중간단계는 크게 문제가 되는 항목은 아니지만, 좀더 효율적으로 동작 할 수 있도록 수정을 권유하는 항목입니다

해당 항목 없음

Low Severity

낮은 위험도는 성능이나 보안에는 문제는 없지만, 코드 가독성, 컨트랙트 구조 개선을 위해
수정을 권유하는 항목입니다.

해당 항목 없음

Conclusion

Bifi 컨트랙트는 **ERC-20** 인터페이스에 맞춰 작성되었으며 검증된 오픈소스인 **Openzeppelin**의 컨트랙트가 주로 참고되었습니다. 주요 기능은 전체 컨트랙트 토큰 전송 정지와 토큰 소각 기능이며 그 외 별도 락업, 추가 발행, 주소 동결 등의 기능은 구현되어 있지않습니다.

토큰 전송 정지 기능은 해킹 및 토큰 탈취와 같은 상황에 있어 상황을 최소화할 수 있으며 메인네트워크 전환에 필요한 토큰 스왑 시 용이하게 동작할 수 있습니다. 해당 기능은 컨트랙트 오너 주요 권한이며 컨트랙트 오너는 추후 권한 포기 기능을 통해 탈 중앙화 형태를 취할 수 있습니다.

컨트랙트의 복잡성이 낮고 간결할 뿐더러 **Openzeppelin**의 검증된 코드를 기반으로 작성되어 있기때문에 보안적 이슈를 발견하지 못하였습니다.

Declare

HEXLANT CONTRACT CERTIFICATION

This contract speicifics that it has been validated by the Hexlant Technical Team and notifies that it has not any technical defects.

PUBLISHIED INFORMATION

REPORT NUMBER	ERC20200602	
DATE	2021/05/10	
PUBLISHIER	Henry	henry@hexlant.com

TOKEN INFORMATION

TOKEN NAME	Bifi		
SYMBOL	Bifi		
PLATFORM	ETHEREUM	**TOKEN TYPE**	ERC-20
TOTAL SUPPLY	1,000,000,000 Bifi		
CONTRACT ADDRESS	0x2791BfD60D232150Bff86b39B7146c0eaAA2BA81		

VULNERABLILLITY ANALYSIS

CRITICAL	0	No relevant provision
HIGH	0	No relevant provision
MEDIUM	0	No relevant provision
LOW	0	No relevant provision

CENTRALIZED FUNCTIONS

FREEZE	No	Ability to freeze tokens in accounts. (The administrator can freeze the hacker's account in case of hacking.)
PAUSE	YES	Ability to pause functions related to token transmission in a contract. (This is used when the administrator needs to prevent the movement of assets due to token swaps or hacking.)
LOCKUP	No	Ability to block token transfers for a period of time (Administrators can use to set lockout periods for investors, team members, advisors, etc.)
BURN	YES	Ability to reduce total supply by burning tokens
MINT	NO	Ability to increase total supply by minting tokens

Certified by Hexlant.

Hexlant.
Blockchain Lab
-
contact@hexlant.com
www.hexlant.com