SOLIDITY.*FINANCE*

Bundles Finance (BUND) - Updated Smart Contract Audit Report

SUMMARY

Bundles allows token holders to use their Crypto prediction skills to choose which cryptocurrencies will perform best over the following 6 days. Out of 10 popular cryptocurrencies, $BUND token holders can choose to stake their tokens on a single asset or a 'Bundle' of assets to achieve the highest returns during the staking period. Depending upon the performance of your $BUND tokens staked in relation to the other $BUND tokens staked over the 6 day period, you will either increase or decrease your token holdings

***Note**:* This is the second audit we have completed for the Bundles team. For this audit we confirmed our prior findings on Bundles' token contracts and analyzed the team's liquidty pool and updated staking contracts. This audit arose after the dev team introduced potentially malicious code into the prior staking contracts after our first audit was complete (details on what happened here). Users should unstake from the old pools and move their tokens to the new pools. At the time of writing this, ~1.5k USD of value remains in the vulnerable contracts, and no malicious actions have been taken or funds lost.

Verified Contracts:

- BUND Token - 0x8d3e855f3f55109d473735ab76f753218400fe96
- BUND + ETH LP Pool - 0x44ef4b1dc8f566350f3cCD5C83c21743151fb98D
- Updated Low-Risk Pool Contract - 0xd1064084f27c1b2c5991f3acc13be8f9916b08cd
- Updated High-Risk Pool Contract - 0x4424682d61bd5d489e202bf242d57f5de09a68b4

There are 3 differences between the high and low risk pools:

- The high risk pool allows users to win/lose up to 40% as opposed to 4% on the low-risk pool. The team controls the system that determines winners; calculate your personal risk tolerance accordingly. The team is public and appears trustworthy.
- The high risk pool's developer fee is 1% of BUND winnings as

Audit Findings:

- *Summary: No issues from outside attackers were identified.*
- *Ensure trust in the project team. If the team acts maliciously the code restricts them from misappropriating more than 4% of user's staked funds in the low-risk pool and 40% in the high-risk pool.*
- *Date: December 5th, 2020.*

- *The BUND token contract is secure and cannot be minted after deployment.*
- *Bundles' development team previously added a drain() function after our audit was complete without informing us. We alterted the community and subsequently helped the team to update their contracts to make them safe. Verify the contracts to be invested in are the same as those verified by our team above.*

- *The logic that manages the ecosystem is run on an off-chain NodeJS server hosted on AWS. The owner-restricted functions of the contracts are called by this server using a privatekey stored on the server in order to determine prices and update user balances. If this AWS account of its owner were compromised, user funds would be at risk.*
- *Mitigation measure: We pointed out this potential issue to the*

funds (Now 40% in the high-risk pool). We have also briefly inspected the NodeJS code and the code appears to be legitimate and serve its intended purpose.

- *While there is risk associated with an owner private key having this control, the actions of the team and their willingness to mitigate this risk makes us believe the team is trustworthy. The team is also publicly known, which further reduces the probability of a malicious owner.*

- *The prices fed to the Oracle contract are sent directly from the contract's owner via a single source (CoinGecko). The data sent to the oracle is not used by the Bundles contract; the team explains this oracle is so users can see the prices used to decide rewards on-chain. If CoinGecko were compromised, user funds would be at risk to an unfair/manipulated outcome.*

COMBINED AUDIT RESULTS

Date: December 5th, 2020

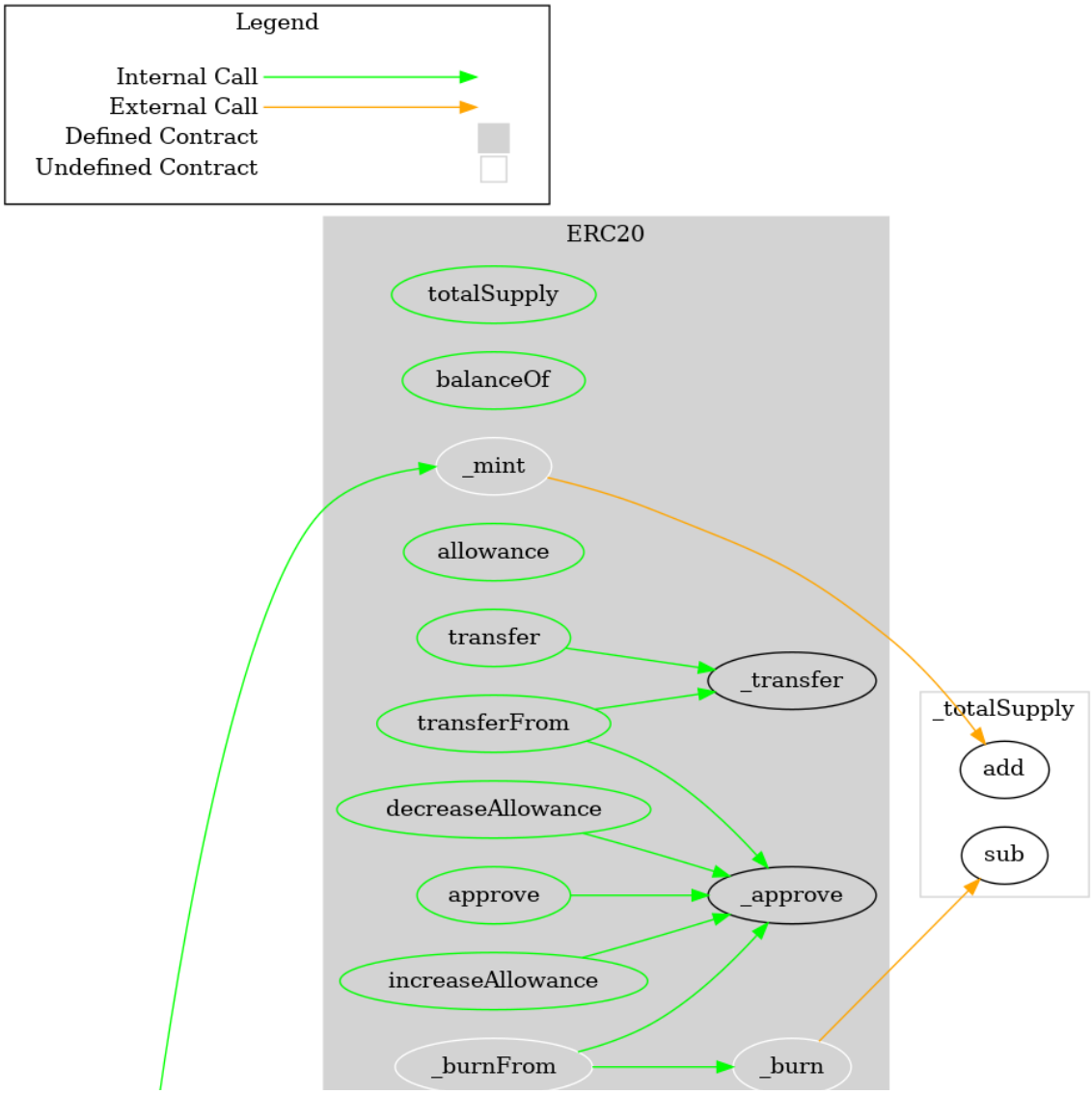
Vulnerability Category	Notes	Result

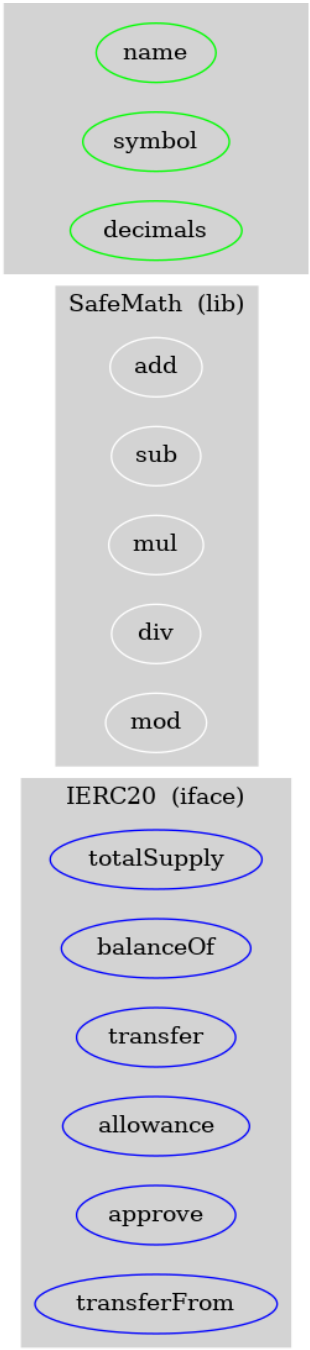
Vulnerability Category	Notes	Result
Arbitrary Jump	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	Some functions rely on predictable environment variables. This is not best practice, but the probability of miners maliciously changing these variables is extremley low.	Warning
Deprecated Opcodes	N/A	PASS
Ether/Token Thief	The owner of the prediciton contract determines and sets the rewards for each user. If the owner key was compromised, a maximum of 4% of each user's staked	Warning

Vulnerability Category	Notes	Result
Exceptions	N/A	PASS
External Calls	N/A	PASS

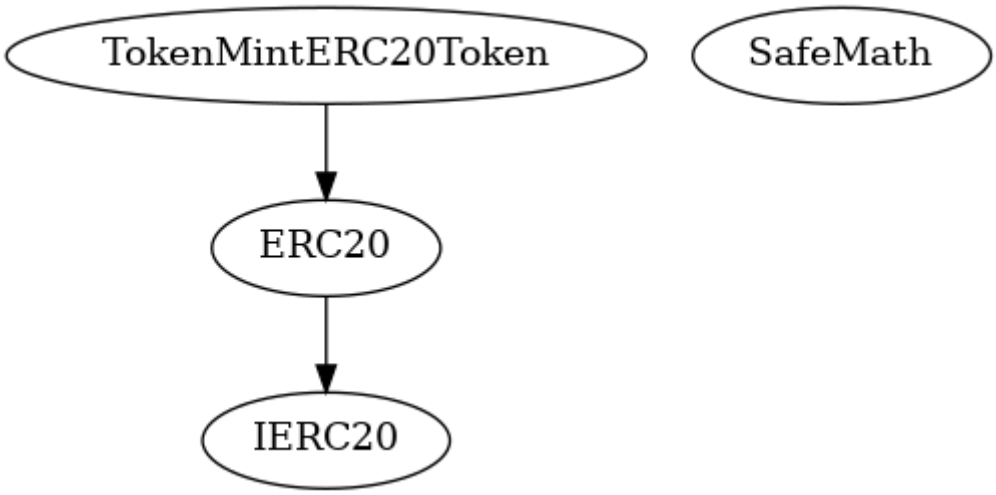
DETAILS: BUNDTOKEN

FUNCTION GRAPH





INHERITENCE CHART



FUNCTIONS OVERVIEW

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

Int = Internal
Ext = External
Pub = Public

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

+ [Lib] SafeMath
```

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

+   ERC20 (IERC20)
    - [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] _burnFrom #

+   TokenMintERC20Token (ERC20)
    - [Pub]  ($)
    - [Pub] burn #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals
```

SOURCE CODE

```
/**
 *Submitted for verification at Etherscan.io on 2019
*/

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;

/**
 * @dev Interface of the ERC20 standard as defined i
 * the optional functions; to access them see `ERC20
 */
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.
```

```
    /**
     * @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
     *
     * > Beware that changing an allowance with this
     * 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
```

```
    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
}

// File: contracts\open-zeppelin-contracts\math\Safe

pragma solidity ^0.5.0;

/**
 * @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
        require(b <= a, "SafeMath: subtraction overf
        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;
        }

        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:
```

```
        // Solidity only automatically asserts when
        require(b > 0, "SafeMath: division by zero")
        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
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;
```

```
 * This implementation is agnostic to the way tokens
 * that a supply mechanism has to be added in a deri
 * For a generic mechanism see `ERC20Mintable`.
 *
 * *For a detailed writeup see our guide [How to imp
 * mechanisms](https://forum.zeppelin.solutions/t/ho
 *
 * 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 IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

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

    uint256 private _totalSupply;

    /**
     * @dev See `IERC20.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(msg.sender, recipient, amount);
        return true;
    }

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

    /**
     * @dev See `IERC20.approve`.
```

```
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 value)
        _approve(msg.sender, spender, value);
        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 `v
     * - the caller must have allowance for `sender`
     * `amount`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sen
        return true;
    }

    /**
     * @dev Atomically increases the allowance grant
     *
     * This is an alternative to `approve` that can
     * problems described in `IERC20.approve`.
     *
```

```
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint
        _approve(msg.sender, spender, _allowances[ms
        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(msg.sender, spender, _allowances[ms
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `
     *
     * This is internal function is equivalent to `t
```

```
     *
     * 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

        _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

        _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 value) i
        require(account != address(0), "ERC20: burn

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

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This is internal function is equivalent to `a
     * 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.
     */
```

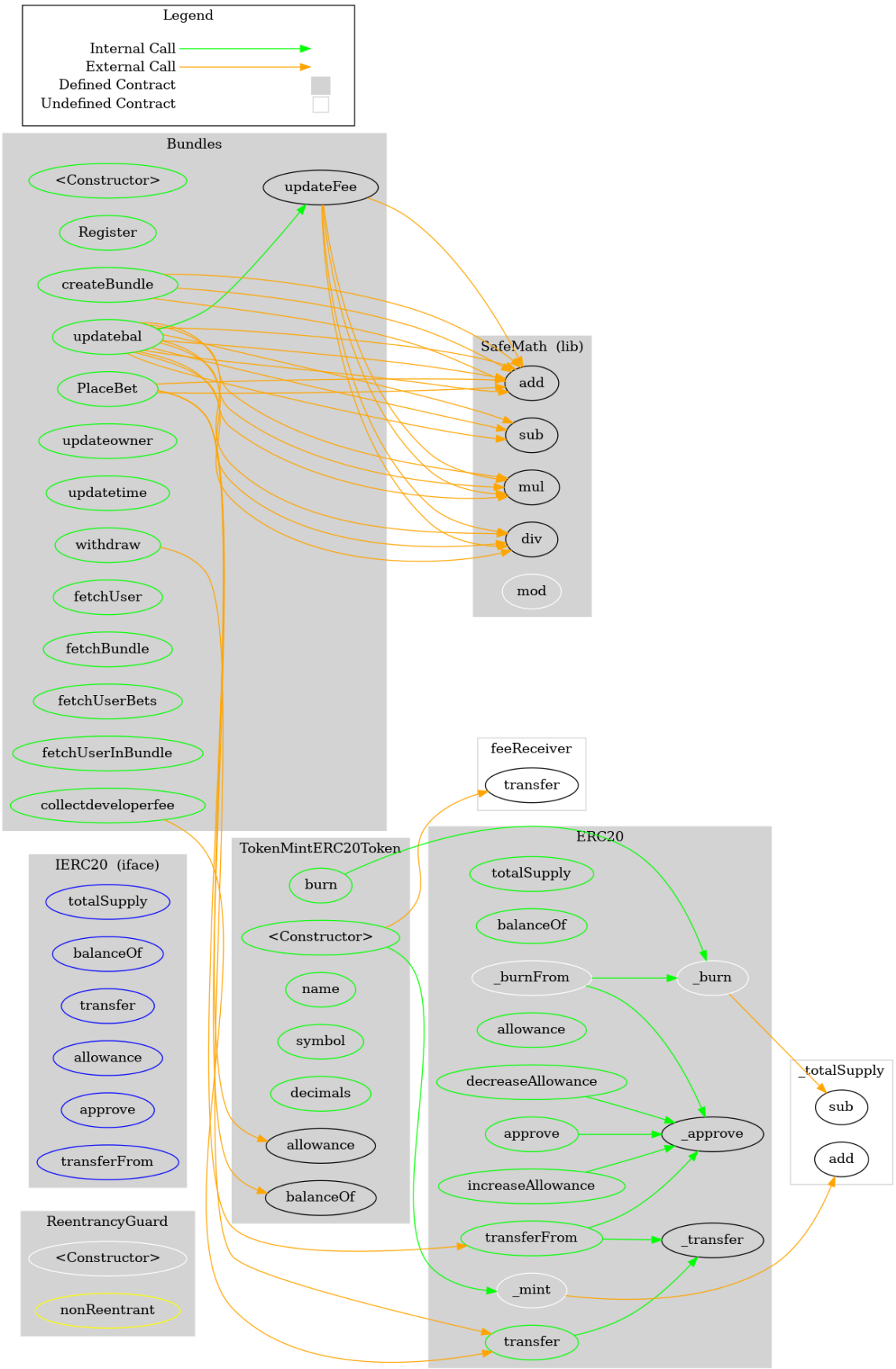
```
        _allowances[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Destoys `amount` tokens from `account`.`
     * from the caller's allowance.
     *
     * See `_burn` and `_approve`.
     */
    function _burnFrom(address account, uint256 amou
        _burn(account, amount);
        _approve(account, msg.sender, _allowances[ac
    }
}

// File: contracts\ERC20\TokenMintERC20Token.sol

pragma solidity ^0.5.0;


/**
 * @title TokenMintERC20Token
 * @author TokenMint (visit https://tokenmint.io)
 *
 * @dev Standard ERC20 token with burning and option
 * For full specification of ERC-20 standard see:
 * https://github.com/ethereum/EIPs/blob/master/EIPS
 */
contract TokenMintERC20Token is ERC20 {
```

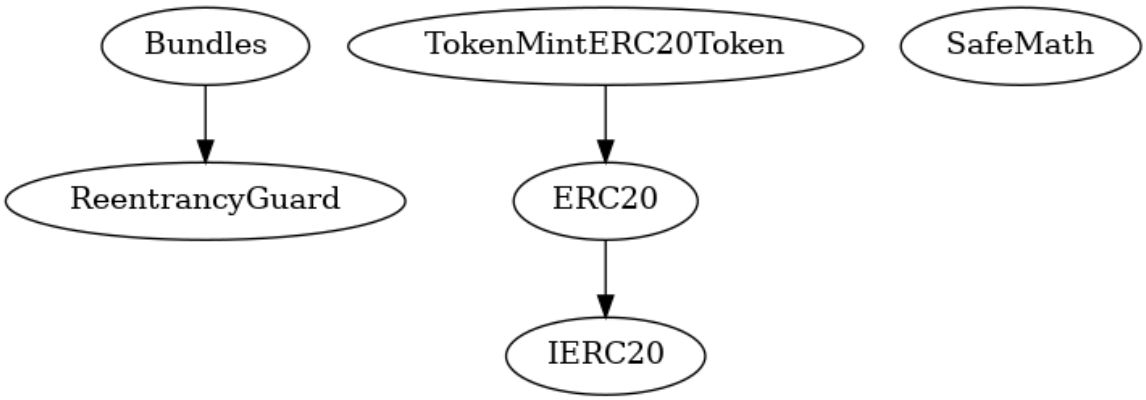
```
    uint8 private _decimals;

    /**
     * @dev Constructor.
     * @param name name of the token
     * @param symbol symbol of the token, 3-4 chars
     * @param decimals number of decimal places of o
     * @param totalSupply total supply of tokens in
     * @param tokenOwnerAddress address that gets 10
     */
    constructor(string memory name, string memory sy
      _name = name;
      _symbol = symbol;
      _decimals = decimals;

      // set tokenOwnerAddress as owner of all token
      _mint(tokenOwnerAddress, totalSupply);

      // pay the service fee for contract deployment
      feeReceiver.transfer(msg.value);
    }

    /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of lowest token units
     */
    function burn(uint256 value) public {
      _burn(msg.sender, value);
    }

    // optional functions from ERC20 stardard
```

```
     */
    function name() public view returns (string memo
      return _name;
    }

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

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8)
      return _decimals;
    }
}
```

DETAILS: BUNDLESV2 (LOW-RISK)

FUNCTION GRAPH





FUNCTIONS OVERVIEW

```
+  ReentrancyGuard
   - [Int]  #

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

+ [Lib] SafeMath
   - [Int] add
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] mod
```

```
    - [Pub] transfer #
    - [Pub] allowance
    - [Pub] approve #
    - [Pub] transferFrom #
    - [Pub] increaseAllowance #
    - [Pub] decreaseAllowance #
    - [Int] _transfer #
    - [Int] _mint #
    - [Int] _burn #
    - [Int] _approve #
    - [Int] _burnFrom #

+   TokenMintERC20Token (ERC20)
    - [Pub]  ($)
    - [Pub] burn #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals

+   Bundles (ReentrancyGuard)
    - [Pub]  #
    - [Pub] Register #
    - [Pub] PlaceBet #
    - [Pub] updatebal #
    - [Int] updateFee #
    - [Pub] createBundle #
    - [Pub] updateowner #
    - [Pub] updatetime #
    - [Pub] withdraw #
       - modifiers: nonReentrant
    - [Pub] fetchUser
```

```
    - [Pub] collectdeveloperfee #
       - modifiers: nonReentrant
```

SOURCE CODE

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

```
// SPDX-License-Identifier: UNLICENSED

pragma solidity <=0.7.5;


// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;

contract ReentrancyGuard {
    // Booleans are more expensive than uint256 or a
    // word because each write operation emits an ex
    // slot's contents, replace the bits taken up by
    // back. This is the compiler's defense against
    // pointer aliasing, and it cannot be disabled.

    // The values being non-zero value makes deploym
    // but in exchange the refund on every call to n
```

```
    uint256 private constant _NOT_ENTERED = 1;
    uint256 private constant _ENTERED = 2;

    uint256 private _status;

    constructor () internal {
        _status = _NOT_ENTERED;
    }

    /**
     * @dev Prevents a contract from calling itself,
     * Calling a `nonReentrant` function from anothe
     * function is not supported. It is possible to
     * by making the `nonReentrant` function externa
     * `private` function that does the actual work.
     */
    modifier nonReentrant() {
        // On the first call to nonReentrant, _notEn
        require(_status != _ENTERED, "ReentrancyGuar

        // Any calls to nonReentrant after this poin
        _status = _ENTERED;

        _;

        // By storing the original value once again,
        // https://eips.ethereum.org/EIPS/eip-2200)
        _status = _NOT_ENTERED;
    }
}
```

```
 */
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

    /**
     * @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

    /**
```

```
     *
     * > Beware that changing an allowance with this
     * 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.
     */
    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
```

```
     */
    event Approval(address indexed owner, address in
}

// File: contracts\open-zeppelin-contracts\math\Safe

pragma solidity ^0.5.0;

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

```
        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
        require(b <= a, "SafeMath: subtraction overf
        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
```

```
            return 0;
        }

        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
        // Solidity only automatically asserts when
        require(b > 0, "SafeMath: division by zero")
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is n

        return c;
    }

    /**
     * @dev Returns the remainder of dividing two un
```

```
     * 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
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;



/**
 * @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 `ERC20Mintable`.
 *
 * *For a detailed writeup see our guide [How to imp
 * mechanisms](https://forum.zeppelin.solutions/t/ho
 *
 * We have followed general OpenZeppelin guidelines:
 * of returning `false` on failure. This behavior is
 * and does not conflict with the expectations of ER
```

```
 * 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 IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

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

    uint256 private _totalSupply;

    /**
     * @dev See `IERC20.totalSupply`.
     */
    function totalSupply() public view returns (uint
        return _totalSupply;
    }

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

    /**
```

```
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least
     */
    function transfer(address recipient, uint256 amo
        _transfer(msg.sender, recipient, amount);
        return true;
    }

    /**
     * @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 value)
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev See `IERC20.transferFrom`.
     *
```

```
     * Requirements:
     * - `sender` and `recipient` cannot be the zero
     * - `sender` must have a balance of at least `v
     * - the caller must have allowance for `sender`
     * `amount`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sen
        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(msg.sender, spender, _allowances[ms
        return true;
    }

    /**
     * @dev Atomically decreases the allowance grant
```

```
     *
     * 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(msg.sender, spender, _allowances[ms
        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
```

```
        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

        _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
     */
```

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

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This is internal function is equivalent to `a
     * 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] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Destoys `amount` tokens from `account`.`
     * from the caller's allowance.
     *
     * See `_burn` and `_approve`.
```

```
        _approve(account, msg.sender, _allowances[ac
    }
}

// File: contracts\ERC20\TokenMintERC20Token.sol

pragma solidity ^0.5.0;


/**
 * @title TokenMintERC20Token
 * @author TokenMint (visit https://tokenmint.io)
 *
 * @dev Standard ERC20 token with burning and option
 * For full specification of ERC-20 standard see:
 * https://github.com/ethereum/EIPs/blob/master/EIPS
 */
contract TokenMintERC20Token is ERC20 {

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

    /**
     * @dev Constructor.
     * @param name name of the token
     * @param symbol symbol of the token, 3-4 chars
     * @param decimals number of decimal places of o
     * @param totalSupply total supply of tokens in
     * @param tokenOwnerAddress address that gets 10
     */
```

```
    _decimals = decimals;

    // set tokenOwnerAddress as owner of all token
    _mint(tokenOwnerAddress, totalSupply);

    // pay the service fee for contract deployment
    feeReceiver.transfer(msg.value);
  }

  /**
   * @dev Burns a specific amount of tokens.
   * @param value The amount of lowest token units
   */
  function burn(uint256 value) public {
    _burn(msg.sender, value);
  }

  // optional functions from ERC20 stardard

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

  /**
   * @return the symbol of the token.
   */
  function symbol() public view returns (string me
    return _symbol;
```

```
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8)
      return _decimals;
    }
}

contract Bundles is ReentrancyGuard {

    uint256 public bundleId = 1;
    address public owner;

    /* Bundle Token Address */

    TokenMintERC20Token public bundle_address;

    /* Variable to store the fee collected */

    uint256 public fee_collected;

    /* Last Created Informations*/

    uint256 public lastcreated;
    uint256 lastbundlecreated;

    struct UserBets{
        uint256[10] bundles;
        uint256[10] amounts;
        uint256[10] prices;
        bool betted;
        uint256 balance;
```

```
struct User{
    uint256[] bundles;
    string username;
    uint256 balance;
    uint256 freebal;
    bool active;
}

struct Data{
    address[] user;
}

struct Bundle{
    uint256[10] prices;
    uint256 startime;
    uint256 stakingends;
    uint256 endtime;
}

mapping(address => mapping(uint256 => UserBets))
mapping(uint256 => Bundle) bundle;
mapping(address => User) user;
mapping(uint256 => Data) data;

constructor(address _bundle_address) public{
    owner = msg.sender;
    bundle_address = TokenMintERC20Token(_bundle
    lastcreated = block.timestamp;
}
```

```
        User storage us = user[msg.sender];
        require(us.active == false,'Existing User');
        us.active = true;
        us.username = _username;
        return true;
    }

    /* For placing a prediction in the bundle. */

    function PlaceBet(uint256 index,uint256 _prices,
        require(_bundleId <= bundleId,'Invalid Bundl
        require(bundle_address.allowance(msg.sender,
        Bundle storage b = bundle[_bundleId];
        Data storage d = data[_bundleId];
        require(b.stakingends >= block.timestamp,'En
        User storage us = user[msg.sender];
        require(us.active == true,'Register to parti
        UserBets storage u = bets[msg.sender][_bundl
        require(u.bundles[index] == 0,'Already Bette
        if(u.betted == false){
            u.balance = bundle_address.balanceOf(msg
            u.betted = true;
        }
        else{
            require(SafeMath.add(u.totalbet,_amount)
        }
        us.bundles.push(_bundleId);
        us.balance = SafeMath.add(us.balance,_amount
        u.bundles[index] = _percent;
        u.prices[index] = _prices;
        u.amounts[index] = _amount;
```

```
        return true;
    }

    /* Update user balance. Max 4% can be changed */

    function updatebal(address _user,uint256 _bundle
        require(msg.sender == owner,'Not Owner');
        require(_reward <= 4000000,'Invalid Reward P
        User storage us = user[_user];
        require(us.active == true,'Invalid User');
        UserBets storage u = bets[_user][_bundleId];
        require(u.claimed == false,'Already Claimed'
        if(_isPositive == true){
            updateFee(_reward,u.totalbet);
            uint256 temp = SafeMath.mul(_reward,90);
            uint256 reward = SafeMath.div(temp,100);
            uint256 a = SafeMath.mul(u.totalbet,rewa
            uint256 b = SafeMath.div(a,10**8);
            uint256 c = SafeMath.add(u.totalbet,b);
            u.claimed = true;
            us.freebal = SafeMath.add(c,us.freebal);
            us.balance = SafeMath.sub(us.balance,u.t
        }
        else{
            uint256 a = SafeMath.mul(u.totalbet,_rew
            uint256 b = SafeMath.div(a,10**8);
            uint256 c = SafeMath.sub(u.totalbet,b);
            u.claimed = true;
            us.freebal = SafeMath.add(c,us.freebal);
            us.balance = SafeMath.sub(us.balance,u.t
        }
```

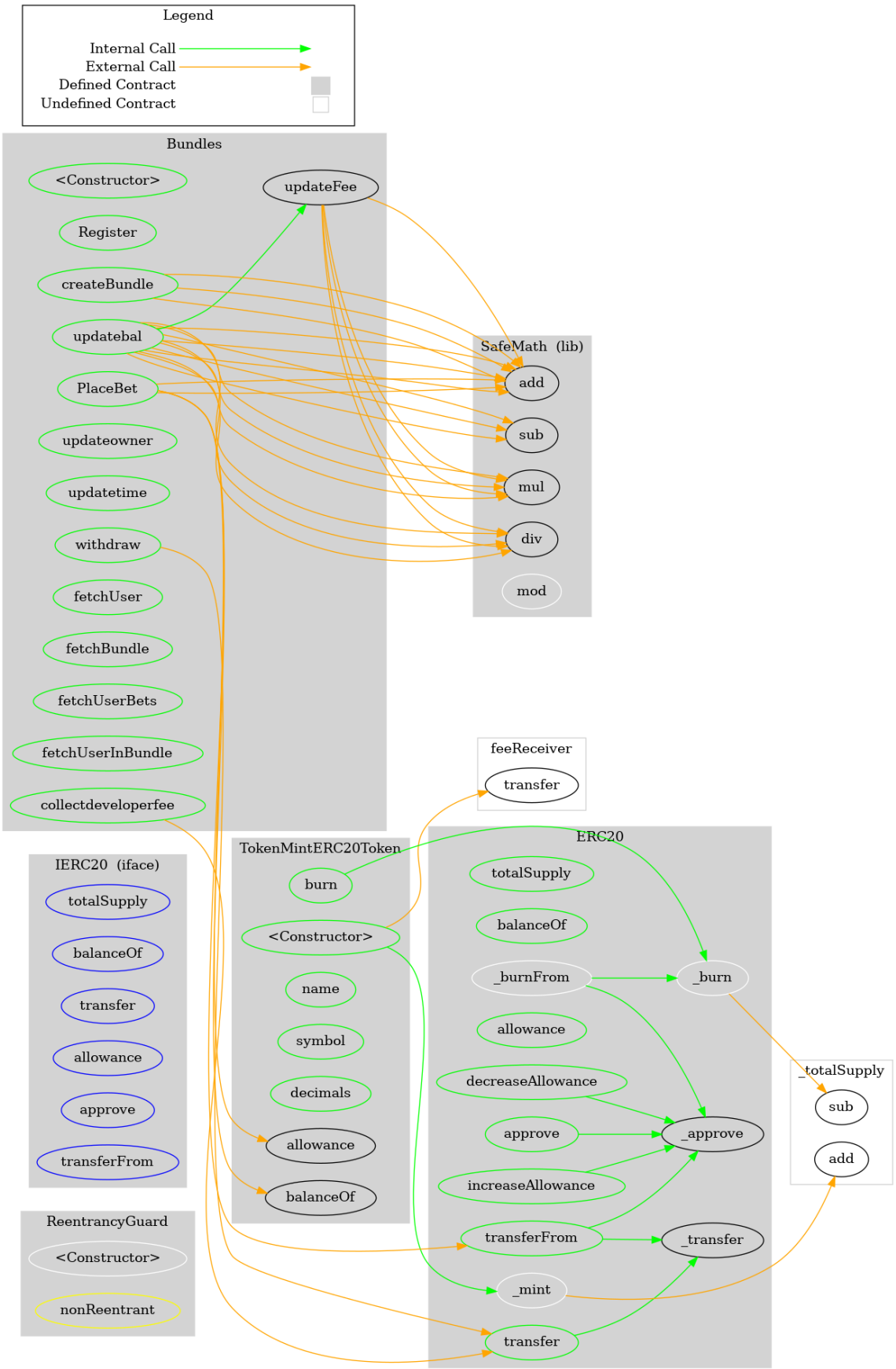
```
    /* Update the fee incurred */

    function updateFee(uint256 r,uint256 amt) intern
        uint256 temp = SafeMath.mul(r,10);
        uint256 reward = SafeMath.div(temp,100);
        uint256 a = SafeMath.mul(amt,reward);
        uint256 b = SafeMath.div(a,10**8);
        fee_collected = SafeMath.add(fee_collected,b
    }

    /* Create a new bundle after 3 days */

    function createBundle(uint256[10] memory _prices
        require(msg.sender == owner,'Not Owner');
        require( block.timestamp > lastbundlecreated
        Bundle storage b = bundle[bundleId];
        b.prices = _prices;
        b.startime = block.timestamp;
        lastbundlecreated = block.timestamp;
        lastcreated = block.timestamp;
        b.endtime = SafeMath.add(block.timestamp,3 d
        b.stakingends = SafeMath.add(block.timestamp
        bundleId = SafeMath.add(bundleId,1);
        return true;
    }

    /* Update new owner of the contract */

    function updateowner(address new_owner) public r
        require(msg.sender == owner,'Not an Owner');
        owner = new_owner;
```

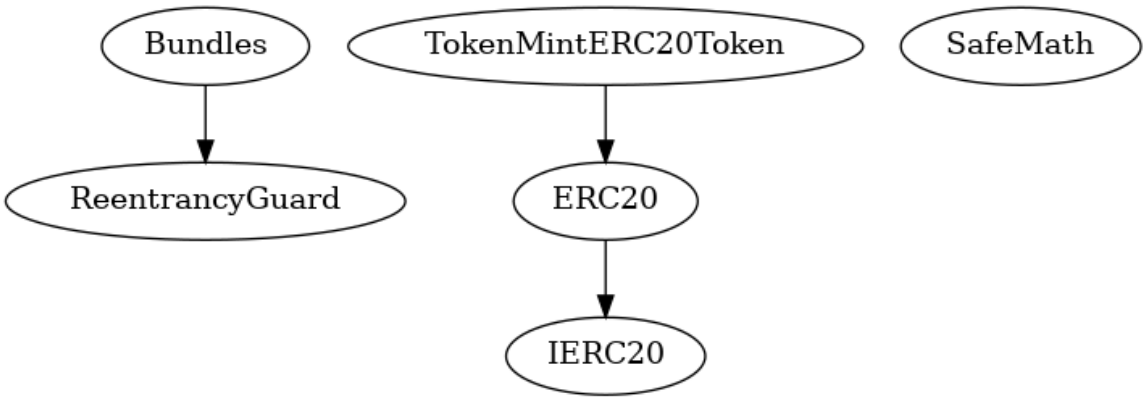
```
    /* Update the timestamp of the last creted bundl

    function updatetime(uint256 _timestamp) public r
        require(msg.sender == owner,'Not an owner');
        lastcreated =  _timestamp;
    }

    /* Allows the user to withdraw his claimable bal

    function withdraw() public nonReentrant returns(
       User storage us = user[msg.sender];
       require(us.active == true,'Invalid User');
       require(us.freebal > 0,'No bal');
       bundle_address.transfer(msg.sender,us.freebal
       us.freebal = 0;
       return true;
    }

    /*  Fetch the information about user. His claima

    function fetchUser(address _user) public view re
        User storage us = user[_user];
        return(us.bundles,us.username,us.freebal,us.
    }

    /* Fetch the information of a BundleId */

    function fetchBundle(uint256 _bundleId) public v
        Bundle storage b = bundle[_bundleId];
        return(b.prices,b.startime,b.endtime,b.staki
    }
```

```
    function fetchUserBets(address _user, uint256 _b
        UserBets storage u = bets[_user][_bundleId];
        return (u.bundles,u.prices,u.amounts,u.balan
    }

    /* Fetch all the user wallet predicted in a bund

    function fetchUserInBundle(uint256 _bundleId) pu
        Data storage d = data[_bundleId];
        return d.user;
    }

    /*
        Only Allow the Developer to withdraw the dev
    */

    function collectdeveloperfee() public nonReentra
        require(msg.sender == owner,'To Be Claimed B
        bundle_address.transfer(msg.sender,fee_colle
        fee_collected = 0;
        return true;
    }

}
```

DETAILS: BUNDLESV2 (HIGH-RISK)

FUNCTION GRAPH





FUNCTIONS OVERVIEW

```
+  ReentrancyGuard
   - [Int]  #

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

+ [Lib] SafeMath
   - [Int] add
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] mod
```

```
    - [Pub] transfer #
    - [Pub] allowance
    - [Pub] approve #
    - [Pub] transferFrom #
    - [Pub] increaseAllowance #
    - [Pub] decreaseAllowance #
    - [Int] _transfer #
    - [Int] _mint #
    - [Int] _burn #
    - [Int] _approve #
    - [Int] _burnFrom #

+   TokenMintERC20Token (ERC20)
    - [Pub]  ($)
    - [Pub] burn #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals

+   Bundles (ReentrancyGuard)
    - [Pub]  #
    - [Pub] Register #
    - [Pub] PlaceBet #
    - [Pub] updatebal #
    - [Int] updateFee #
    - [Pub] createBundle #
    - [Pub] updateowner #
    - [Pub] updatetime #
    - [Pub] withdraw #
       - modifiers: nonReentrant
    - [Pub] fetchUser
```

```
    - [Pub] collectdeveloperfee #
       - modifiers: nonReentrant
```

SOURCE CODE

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

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

// SPDX-License-Identifier: UNLICENSED

pragma solidity <=0.7.5;


// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;

contract ReentrancyGuard {
    // Booleans are more expensive than uint256 or a
    // word because each write operation emits an ex
    // slot's contents, replace the bits taken up by
    // back. This is the compiler's defense against
```

```
    // but in exchange the refund on every call to n
    // amount. Since refunds are capped to a percent
    // transaction's gas, it is best to keep them lo
    // increase the likelihood of the full refund co
    uint256 private constant _NOT_ENTERED = 1;
    uint256 private constant _ENTERED = 2;

    uint256 private _status;

    constructor () internal {
        _status = _NOT_ENTERED;
    }

    /**
     * @dev Prevents a contract from calling itself,
     * Calling a `nonReentrant` function from anothe
     * function is not supported. It is possible to
     * by making the `nonReentrant` function externa
     * `private` function that does the actual work.
     */
    modifier nonReentrant() {
        // On the first call to nonReentrant, _notEn
        require(_status != _ENTERED, "ReentrancyGuar

        // Any calls to nonReentrant after this poin
        _status = _ENTERED;

        _;

        // By storing the original value once again,
        // https://eips.ethereum.org/EIPS/eip-2200)
```

```
/**
 * @dev Interface of the ERC20 standard as defined i
 * the optional functions; to access them see `ERC20
 */
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

    /**
     * @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
```

```
    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * Returns a boolean value indicating whether th
     *
     * > Beware that changing an allowance with this
     * 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.
     */
    function transferFrom(address sender, address re

    /**
     * @dev Emitted when `value` tokens are moved fr
     * another (`to`).
     *
```

```
    /**
     * @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: contracts\open-zeppelin-contracts\math\Safe

pragma solidity ^0.5.0;

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

```
     */
    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
        require(b <= a, "SafeMath: subtraction overf
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Returns the multiplication of two unsign
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
```

```
        // 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;
        }

        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
        // Solidity only automatically asserts when
        require(b > 0, "SafeMath: division by zero")
        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
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;



/**
 * @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 `ERC20Mintable`.
 *
 * *For a detailed writeup see our guide [How to imp
 * mechanisms](https://forum.zeppelin.solutions/t/ho
```

```
 * 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 IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

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

    uint256 private _totalSupply;

    /**
     * @dev See `IERC20.totalSupply`.
     */
    function totalSupply() public view returns (uint
        return _totalSupply;
    }

    /**
     * @dev See `IERC20.balanceOf`.
     */
    function balanceOf(address account) public view
```

```
    /**
     * @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(msg.sender, recipient, amount);
        return true;
    }

    /**
     * @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 value)
        _approve(msg.sender, spender, value);
        return true;
    }
```

```
     *
     * 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 `v
     * - the caller must have allowance for `sender`
     * `amount`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sen
        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(msg.sender, spender, _allowances[ms
        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(msg.sender, spender, _allowances[ms
        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
```

```
        require(recipient != address(0), "ERC20: tra

        _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

        _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
```

```
     */
    function _burn(address account, uint256 value) i
        require(account != address(0), "ERC20: burn

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

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * This is internal function is equivalent to `a
     * 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] = value;
        emit Approval(owner, spender, value);
    }

    /**
```

```
     * See `_burn` and `_approve`.
     */
    function _burnFrom(address account, uint256 amou
        _burn(account, amount);
        _approve(account, msg.sender, _allowances[ac
    }
}

// File: contracts\ERC20\TokenMintERC20Token.sol

pragma solidity ^0.5.0;


/**
 * @title TokenMintERC20Token
 * @author TokenMint (visit https://tokenmint.io)
 *
 * @dev Standard ERC20 token with burning and option
 * For full specification of ERC-20 standard see:
 * https://github.com/ethereum/EIPs/blob/master/EIPS
 */
contract TokenMintERC20Token is ERC20 {

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

    /**
     * @dev Constructor.
     * @param name name of the token
     * @param symbol symbol of the token, 3-4 chars
```

```
 */
constructor(string memory name, string memory sy
  _name = name;
  _symbol = symbol;
  _decimals = decimals;

  // set tokenOwnerAddress as owner of all token
  _mint(tokenOwnerAddress, totalSupply);

  // pay the service fee for contract deployment
  feeReceiver.transfer(msg.value);
}

/**
 * @dev Burns a specific amount of tokens.
 * @param value The amount of lowest token units
 */
function burn(uint256 value) public {
  _burn(msg.sender, value);
}

// optional functions from ERC20 stardard

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

/**
```

```
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8)
        return _decimals;
    }
}

contract Bundles is ReentrancyGuard {

    uint256 public bundleId = 1;
    address public owner;

    /* Bundle Token Address */

    TokenMintERC20Token public bundle_address;

    /* Variable to store the fee collected */

    uint256 public fee_collected;

    /* Last Created Informations*/

    uint256 public lastcreated;
    uint256 lastbundlecreated;

    struct UserBets{
        uint256[10] bundles;
```

```
        uint256 balance;
        uint256 totalbet;
        bool claimed;
    }

    struct User{
        uint256[] bundles;
        string username;
        uint256 balance;
        uint256 freebal;
        bool active;
    }

    struct Data{
        address[] user;
    }

    struct Bundle{
        uint256[10] prices;
        uint256 startime;
        uint256 stakingends;
        uint256 endtime;
    }

    mapping(address => mapping(uint256 => UserBets))
    mapping(uint256 => Bundle) bundle;
    mapping(address => User) user;
    mapping(uint256 => Data) data;

    constructor(address _bundle_address) public{
        owner = msg.sender;
```

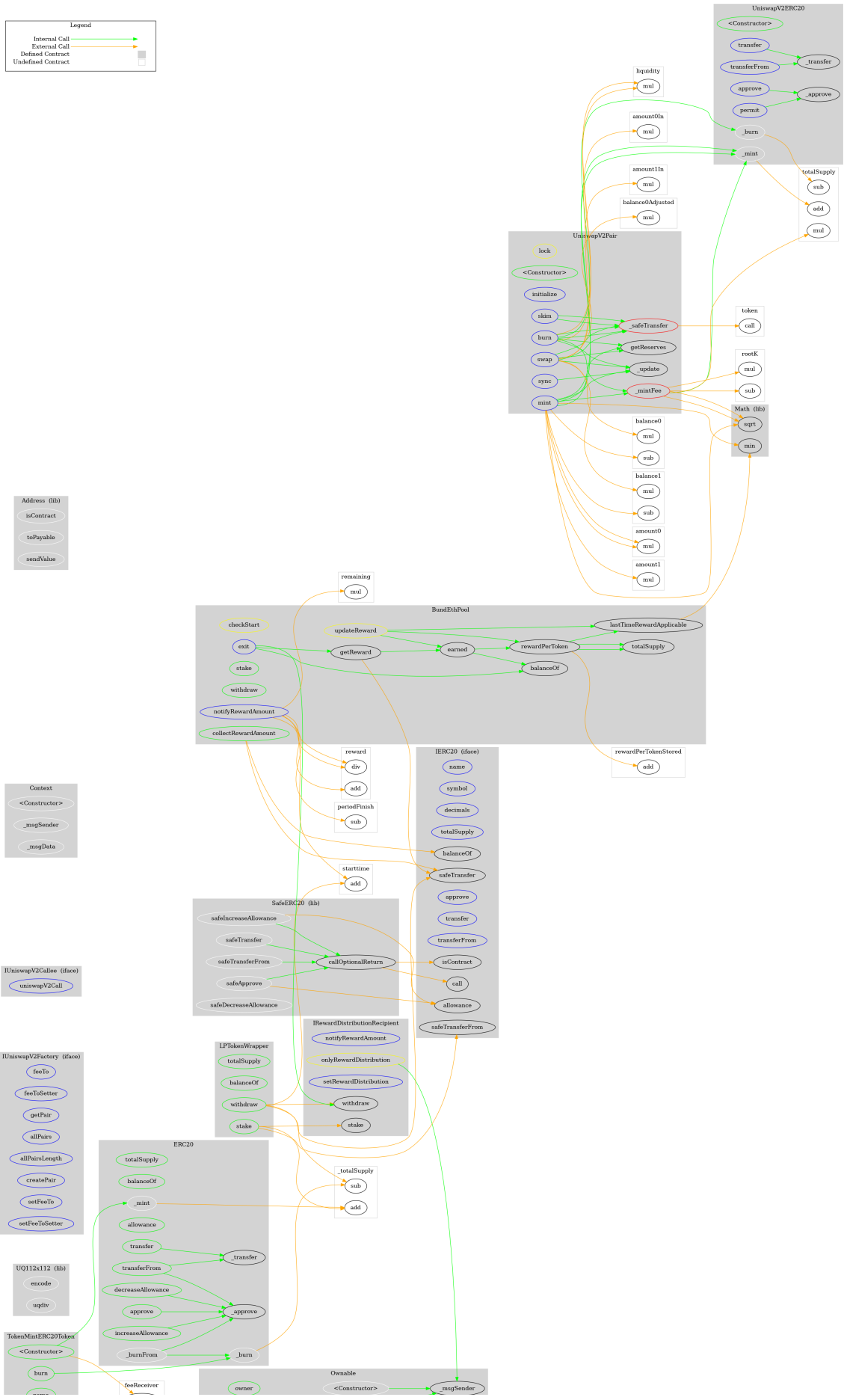
```
    /*  Registering the username to the contract */

    function Register(string memory _username) publi
        User storage us = user[msg.sender];
        require(us.active == false,'Existing User');
        us.active = true;
        us.username = _username;
        return true;
    }

    /* For placing a prediction in the bundle. */

    function PlaceBet(uint256 index,uint256 _prices,
        require(_bundleId <= bundleId,'Invalid Bundl
        require(bundle_address.allowance(msg.sender,
        Bundle storage b = bundle[_bundleId];
        Data storage d = data[_bundleId];
        require(b.stakingends >= block.timestamp,'En
        User storage us = user[msg.sender];
        require(us.active == true,'Register to parti
        UserBets storage u = bets[msg.sender][_bundl
        require(u.bundles[index] == 0,'Already Bette
        if(u.betted == false){
            u.balance = bundle_address.balanceOf(msg
            u.betted = true;
        }
        else{
            require(SafeMath.add(u.totalbet,_amount)
        }
        us.bundles.push(_bundleId);
```

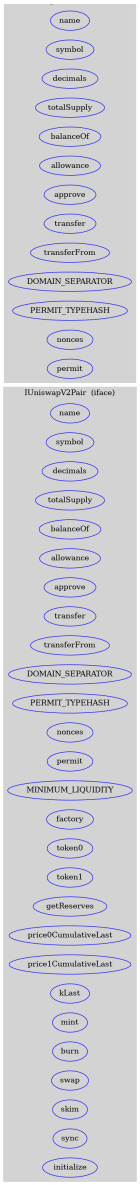
```
        u.amounts[index] = _amount;
        u.totalbet = u.totalbet + _amount;
        d.user.push(msg.sender);
        bundle_address.transferFrom(msg.sender,addre
        return true;
    }

    /* Update user balance. Max 40% can be changed *

    function updatebal(address _user,uint256 _bundle
        require(msg.sender == owner,'Not Owner');
        require(_reward <= 40000000,'Invalid Reward
        User storage us = user[_user];
        require(us.active == true,'Invalid User');
        UserBets storage u = bets[_user][_bundleId];
        require(u.claimed == false,'Already Claimed'
        if(_isPositive == true){
            updateFee(_reward,u.totalbet);
            uint256 temp = SafeMath.mul(_reward,90);
            uint256 reward = SafeMath.div(temp,100);
            uint256 a = SafeMath.mul(u.totalbet,rewa
            uint256 b = SafeMath.div(a,10**8);
            uint256 c = SafeMath.add(u.totalbet,b);
            u.claimed = true;
            us.freebal = SafeMath.add(c,us.freebal);
            us.balance = SafeMath.sub(us.balance,u.t
        }
        else{
            uint256 a = SafeMath.mul(u.totalbet,_rew
            uint256 b = SafeMath.div(a,10**8);
            uint256 c = SafeMath.sub(u.totalbet,b);
```

```
        }
        return true;
    }

    /* Update the fee incurred */

    function updateFee(uint256 r,uint256 amt) intern
        uint256 temp = SafeMath.mul(r,1);
        uint256 reward = SafeMath.div(temp,100);
        uint256 a = SafeMath.mul(amt,reward);
        uint256 b = SafeMath.div(a,10**8);
        fee_collected = SafeMath.add(fee_collected,b
    }

    /* Create a new bundle after 3 days */

    function createBundle(uint256[10] memory _prices
        require(msg.sender == owner,'Not Owner');
        require( block.timestamp > lastbundlecreated
        Bundle storage b = bundle[bundleId];
        b.prices = _prices;
        b.startime = block.timestamp;
        lastbundlecreated = block.timestamp;
        lastcreated = block.timestamp;
        b.endtime = SafeMath.add(block.timestamp,2 d
        b.stakingends = SafeMath.add(block.timestamp
        bundleId = SafeMath.add(bundleId,1);
        return true;
    }

    /* Update new owner of the contract */
```

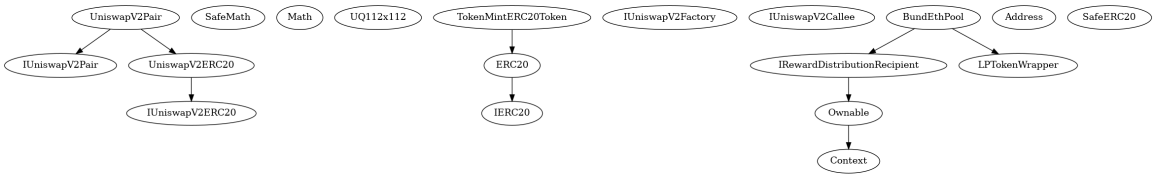
```
        owner = new_owner;
        return true;
    }

    /* Update the timestamp of the last creted bundl

    function updatetime(uint256 _timestamp) public r
        require(msg.sender == owner,'Not an owner');
        lastcreated =  _timestamp;
    }

    /* Allows the user to withdraw his claimable bal

    function withdraw() public nonReentrant returns(
       User storage us = user[msg.sender];
       require(us.active == true,'Invalid User');
       require(us.freebal > 0,'No bal');
       bundle_address.transfer(msg.sender,us.freebal
       us.freebal = 0;
       return true;
    }

    /*  Fetch the information about user. His claima

    function fetchUser(address _user) public view re
        User storage us = user[_user];
        return(us.bundles,us.username,us.freebal,us.
    }

    /* Fetch the information of a BundleId */
```

```
    }

    /* Fetch the prediction information of each user

    function fetchUserBets(address _user, uint256 _b
        UserBets storage u = bets[_user][_bundleId];
        return (u.bundles,u.prices,u.amounts,u.balan
    }

    /* Fetch all the user wallet predicted in a bund

    function fetchUserInBundle(uint256 _bundleId) pu
        Data storage d = data[_bundleId];
        return d.user;
    }

    /*
        Only Allow the Developer to withdraw the dev
    */

    function collectdeveloperfee() public nonReentra
        require(msg.sender == owner,'To Be Claimed B
        bundle_address.transfer(msg.sender,fee_colle
        fee_collected = 0;
        return true;
    }

}
```





INHERITENCE CHART



FUNCTIONS OVERVIEW

```
+ [Int] IUniswapV2Pair
   - [Ext] name
   - [Ext] symbol
   - [Ext] decimals
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transfer #
   - [Ext] transferFrom #
   - [Ext] DOMAIN_SEPARATOR
   - [Ext] PERMIT_TYPEHASH
   - [Ext] nonces
   - [Ext] permit #
   - [Ext] MINIMUM_LIQUIDITY
   - [Ext] factory
   - [Ext] token0
   - [Ext] token1
   - [Ext] getReserves
   - [Ext] price0CumulativeLast
   - [Ext] price1CumulativeLast
   - [Ext] kLast
   - [Ext] mint #
   - [Ext] burn #
   - [Ext] swap #
   - [Ext] skim #
   - [Ext] sync #
   - [Ext] initialize #

+ [Int] IUniswapV2ERC20
   - [Ext] name
```

```
    - [Ext] balanceOf
    - [Ext] allowance
    - [Ext] approve #
    - [Ext] transfer #
    - [Ext] transferFrom #
    - [Ext] DOMAIN_SEPARATOR
    - [Ext] PERMIT_TYPEHASH
    - [Ext] nonces
    - [Ext] permit #

+ [Lib] SafeMath
    - [Int] add
    - [Int] sub
    - [Int] mul
    - [Int] div

+  UniswapV2ERC20 (IUniswapV2ERC20)
    - [Pub]  #
    - [Int] _mint #
    - [Int] _burn #
    - [Prv] _approve #
    - [Prv] _transfer #
    - [Ext] approve #
    - [Ext] transfer #
    - [Ext] transferFrom #
    - [Ext] permit #

+ [Lib] Math
    - [Int] min
    - [Int] sqrt
```

```
+ [Int] IERC20
   - [Ext] name
   - [Ext] symbol
   - [Ext] decimals
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transfer #
   - [Ext] transferFrom #

+ [Int] IUniswapV2Factory
   - [Ext] feeTo
   - [Ext] feeToSetter
   - [Ext] getPair
   - [Ext] allPairs
   - [Ext] allPairsLength
   - [Ext] createPair #
   - [Ext] setFeeTo #
   - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Callee
   - [Ext] uniswapV2Call #

+  UniswapV2Pair (IUniswapV2Pair, UniswapV2ERC20)
   - [Pub] getReserves
   - [Prv] _safeTransfer #
   - [Pub]  #
   - [Ext] initialize #
   - [Prv] _update #
```

```
    - [Ext] burn #
       - modifiers: lock
    - [Ext] swap #
       - modifiers: lock
    - [Ext] skim #
       - modifiers: lock
    - [Ext] sync #
       - modifiers: lock

+  ERC20 (IERC20)
    - [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] _burnFrom #

+  TokenMintERC20Token (ERC20)
    - [Pub]  ($)
    - [Pub] burn #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals
```

```
   - [Int] _msgData

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

+ [Lib] Address
   - [Int] isContract
   - [Int] toPayable
   - [Int] sendValue #

+ [Lib] SafeERC20
   - [Int] safeTransfer #
   - [Int] safeTransferFrom #
   - [Int] safeApprove #
   - [Int] safeIncreaseAllowance #
   - [Int] safeDecreaseAllowance #
   - [Prv] callOptionalReturn #

+  IRewardDistributionRecipient (Ownable)
   - [Ext] notifyRewardAmount #
   - [Ext] setRewardDistribution #
      - modifiers: onlyOwner

+  LPTokenWrapper
```

```
    - [Pub] withdraw #

 +  BundEthPool (LPTokenWrapper, IRewardDistribution
    - [Pub] lastTimeRewardApplicable
    - [Pub] rewardPerToken
    - [Pub] earned
    - [Pub] collectRewardAmount #
       - modifiers: onlyOwner
    - [Pub] stake #
       - modifiers: updateReward,checkStart
    - [Pub] withdraw #
       - modifiers: updateReward,checkStart
    - [Ext] exit #
    - [Pub] getReward #
       - modifiers: updateReward,checkStart
    - [Ext] notifyRewardAmount #
       - modifiers: onlyRewardDistribution,updateRew
```

SOURCE CODE

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

```
// File: contracts/interfaces/IUniswapV2Pair.sol
```

```
    event Approval(address indexed owner, address in
    event Transfer(address indexed from, address ind

    function name() external pure returns (string me
    function symbol() external pure returns (string
    function decimals() external pure returns (uint8
    function totalSupply() external view returns (ui
    function balanceOf(address owner) external view
    function allowance(address owner, address spende

    function approve(address spender, uint value) ex
    function transfer(address to, uint value) extern
    function transferFrom(address from, address to,

    function DOMAIN_SEPARATOR() external view return
    function PERMIT_TYPEHASH() external pure returns
    function nonces(address owner) external view ret

    function permit(address owner, address spender,

    event Mint(address indexed sender, uint amount0,
    event Burn(address indexed sender, uint amount0,
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);
```

```
    function token0() external view returns (address
    function token1() external view returns (address
    function getReserves() external view returns (ui
    function price0CumulativeLast() external view re
    function price1CumulativeLast() external view re
    function kLast() external view returns (uint);

    function mint(address to) external returns (uint
    function burn(address to) external returns (uint
    function swap(uint amount0Out, uint amount1Out,
    function skim(address to) external;
    function sync() external;

    function initialize(address, address) external;
}

// File: contracts/interfaces/IUniswapV2ERC20.sol

pragma solidity >=0.5.0;

interface IUniswapV2ERC20 {
    event Approval(address indexed owner, address in
    event Transfer(address indexed from, address ind

    function name() external pure returns (string me
    function symbol() external pure returns (string
    function decimals() external pure returns (uint8
    function totalSupply() external view returns (ui
    function balanceOf(address owner) external view
    function allowance(address owner, address spende
```

```
    function DOMAIN_SEPARATOR() external view return
    function PERMIT_TYPEHASH() external pure returns
    function nonces(address owner) external view ret

    function permit(address owner, address spender,
}

// File: contracts/libraries/SafeMath.sol

pragma solidity =0.5.16;

// a library for performing overflow-safe math, cour

library SafeMath {
    function add(uint x, uint y) internal pure retur
        require((z = x + y) >= x, 'ds-math-add-overf
    }

    function sub(uint x, uint y) internal pure retur
        require((z = x - y) <= x, 'ds-math-sub-under
    }

    function mul(uint x, uint y) internal pure retur
        require(y == 0 || (z = x * y) / y == x, 'ds-
    }

    function div(uint256 a, uint256 b) internal pure
        return div(a, b);
    }
}
```

```
pragma solidity =0.5.16;



contract UniswapV2ERC20 is IUniswapV2ERC20 {
    using SafeMath for uint;

    string public constant name = 'Uniswap V2';
    string public constant symbol = 'UNI-V2';
    uint8 public constant decimals = 18;
    uint  public totalSupply;
    mapping(address => uint) public balanceOf;
    mapping(address => mapping(address => uint)) pub

    bytes32 public DOMAIN_SEPARATOR;
    // keccak256("Permit(address owner,address spend
    bytes32 public constant PERMIT_TYPEHASH = 0x6e71
    mapping(address => uint) public nonces;

    event Approval(address indexed owner, address in
    event Transfer(address indexed from, address ind

    constructor() public {
        uint chainId;
        assembly {
            chainId := chainid
        }
        DOMAIN_SEPARATOR = keccak256(
            abi.encode(
                keccak256('EIP712Domain(string name,
                keccak256(bytes(name)),
```

```
            )
        );
    }

    function _mint(address to, uint value) internal
        totalSupply = totalSupply.add(value);
        balanceOf[to] = balanceOf[to].add(value);
        emit Transfer(address(0), to, value);
    }

    function _burn(address from, uint value) interna
        balanceOf[from] = balanceOf[from].sub(value)
        totalSupply = totalSupply.sub(value);
        emit Transfer(from, address(0), value);
    }

    function _approve(address owner, address spender
        allowance[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    function _transfer(address from, address to, uin
        balanceOf[from] = balanceOf[from].sub(value)
        balanceOf[to] = balanceOf[to].add(value);
        emit Transfer(from, to, value);
    }

    function approve(address spender, uint value) ex
        _approve(msg.sender, spender, value);
        return true;
    }
```

```
        return true;
    }

    function transferFrom(address from, address to,
        if (allowance[from][msg.sender] != uint(-1))
            allowance[from][msg.sender] = allowance[
        }
        _transfer(from, to, value);
        return true;
    }

    function permit(address owner, address spender,
        require(deadline >= block.timestamp, 'Uniswa
        bytes32 digest = keccak256(
            abi.encodePacked(
                '\x19\x01',
                DOMAIN_SEPARATOR,
                keccak256(abi.encode(PERMIT_TYPEHASH
            )
        );
        address recoveredAddress = ecrecover(digest,
        require(recoveredAddress != address(0) && re
        _approve(owner, spender, value);
    }
}

// File: contracts/libraries/Math.sol

pragma solidity =0.5.16;

// a library for performing various math operations
```

```
        z = x < y ? x : y;
    }

    // babylonian method (https://en.wikipedia.org/w
    function sqrt(uint y) internal pure returns (uin
        if (y > 3) {
            z = y;
            uint x = y / 2 + 1;
            while (x < z) {
                z = x;
                x = (y / x + x) / 2;
            }
        } else if (y != 0) {
            z = 1;
        }
    }
}

// File: contracts/libraries/UQ112x112.sol

pragma solidity =0.5.16;

// a library for handling binary fixed point numbers

// range: [0, 2**112 - 1]
// resolution: 1 / 2**112

library UQ112x112 {
    uint224 constant Q112 = 2**112;

    // encode a uint112 as a UQ112x112
```

```
    // divide a UQ112x112 by a uint112, returning a
    function uqdiv(uint224 x, uint112 y) internal pu
        z = x / uint224(y);
    }
}

// File: contracts/interfaces/IERC20.sol

pragma solidity >=0.5.0;

interface IERC20 {
    event Approval(address indexed owner, address in
    event Transfer(address indexed from, address ind

    function name() external view returns (string me
    function symbol() external view returns (string
    function decimals() external view returns (uint8
    function totalSupply() external view returns (ui
    function balanceOf(address owner) external view
    function allowance(address owner, address spende

    function approve(address spender, uint value) ex
    function transfer(address to, uint value) extern
    function transferFrom(address from, address to,
}

// File: contracts/interfaces/IUniswapV2Factory.sol

pragma solidity >=0.5.0;
```

```
    function feeTo() external view returns (address)
    function feeToSetter() external view returns (ad

    function getPair(address tokenA, address tokenB)
    function allPairs(uint) external view returns (a
    function allPairsLength() external view returns

    function createPair(address tokenA, address toke

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;
}

// File: contracts/interfaces/IUniswapV2Callee.sol

pragma solidity >=0.5.0;

interface IUniswapV2Callee {
    function uniswapV2Call(address sender, uint amou
}

// File: contracts/UniswapV2Pair.sol

pragma solidity =0.5.16;
```

```
    using UQ112x112 for uint224;

    uint public constant MINIMUM_LIQUIDITY = 10**3;
    bytes4 private constant SELECTOR = bytes4(keccak

    address public factory;
    address public token0;
    address public token1;

    uint112 private reserve0;           // uses sing
    uint112 private reserve1;           // uses sing
    uint32  private blockTimestampLast; // uses sing

    uint public price0CumulativeLast;
    uint public price1CumulativeLast;
    uint public kLast; // reserve0 * reserve1, as of

    uint private unlocked = 1;
    modifier lock() {
        require(unlocked == 1, 'UniswapV2: LOCKED');
        unlocked = 0;
        _;
        unlocked = 1;
    }

    function getReserves() public view returns (uint
        _reserve0 = reserve0;
        _reserve1 = reserve1;
        _blockTimestampLast = blockTimestampLast;
    }
```

```
    }

    event Mint(address indexed sender, uint amount0,
    event Burn(address indexed sender, uint amount0,
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    constructor() public {
        factory = msg.sender;
    }

    // called once by the factory at time of deploym
    function initialize(address _token0, address _to
        require(msg.sender == factory, 'UniswapV2: F
        token0 = _token0;
        token1 = _token1;
    }

    // update reserves and, on the first call per bl
    function _update(uint balance0, uint balance1, u
        require(balance0 <= uint112(-1) && balance1
        uint32 blockTimestamp = uint32(block.timesta
        uint32 timeElapsed = blockTimestamp - blockT
        if (timeElapsed > 0 && _reserve0 != 0 && _re
```

```
        }
        reserve0 = uint112(balance0);
        reserve1 = uint112(balance1);
        blockTimestampLast = blockTimestamp;
        emit Sync(reserve0, reserve1);
    }

    // if fee is on, mint liquidity equivalent to 1/
    function _mintFee(uint112 _reserve0, uint112 _re
        address feeTo = IUniswapV2Factory(factory).f
        feeOn = feeTo != address(0);
        uint _kLast = kLast; // gas savings
        if (feeOn) {
            if (_kLast != 0) {
                uint rootK = Math.sqrt(uint(_reserve
                uint rootKLast = Math.sqrt(_kLast);
                if (rootK > rootKLast) {
                    uint numerator = totalSupply.mul
                    uint denominator = rootK.mul(5).
                    uint liquidity = numerator / den
                    if (liquidity > 0) _mint(feeTo,
                }
            }
        } else if (_kLast != 0) {
            kLast = 0;
        }
    }

    // this low-level function should be called from
    function mint(address to) external lock returns
        (uint112 _reserve0, uint112 _reserve1,) = ge
```

```
        uint amount1 = balance1.sub(_reserve1);

        bool feeOn = _mintFee(_reserve0, _reserve1);
        uint _totalSupply = totalSupply; // gas savi
        if (_totalSupply == 0) {
            liquidity = Math.sqrt(amount0.mul(amount
           _mint(address(0), MINIMUM_LIQUIDITY); //
        } else {
            liquidity = Math.min(amount0.mul(_totalS
        }
        require(liquidity > 0, 'UniswapV2: INSUFFICI
        _mint(to, liquidity);

        _update(balance0, balance1, _reserve0, _rese
        if (feeOn) kLast = uint(reserve0).mul(reserv
        emit Mint(msg.sender, amount0, amount1);
    }

    // this low-level function should be called from
    function burn(address to) external lock returns
        (uint112 _reserve0, uint112 _reserve1,) = ge
        address _token0 = token0;
        address _token1 = token1;
        uint balance0 = IERC20(_token0).balanceOf(ad
        uint balance1 = IERC20(_token1).balanceOf(ad
        uint liquidity = balanceOf[address(this)];

        bool feeOn = _mintFee(_reserve0, _reserve1);
        uint _totalSupply = totalSupply; // gas savi
        amount0 = liquidity.mul(balance0) / _totalSu
        amount1 = liquidity.mul(balance1) / _totalSu
```

```
        _safeTransfer(_token1, to, amount1);
        balance0 = IERC20(_token0).balanceOf(address
        balance1 = IERC20(_token1).balanceOf(address

        _update(balance0, balance1, _reserve0, _rese
        if (feeOn) kLast = uint(reserve0).mul(reserv
        emit Burn(msg.sender, amount0, amount1, to);
    }

    // this low-level function should be called from
    function swap(uint amount0Out, uint amount1Out,
        require(amount0Out > 0 || amount1Out > 0, 'U
        (uint112 _reserve0, uint112 _reserve1,) = ge
        require(amount0Out < _reserve0 && amount1Out

        uint balance0;
        uint balance1;
        { // scope for _token{0,1}, avoids stack too
        address _token0 = token0;
        address _token1 = token1;
        require(to != _token0 && to != _token1, 'Uni
        if (amount0Out > 0) _safeTransfer(_token0, t
        if (amount1Out > 0) _safeTransfer(_token1, t
        if (data.length > 0) IUniswapV2Callee(to).un
        balance0 = IERC20(_token0).balanceOf(address
        balance1 = IERC20(_token1).balanceOf(address
        }
        uint amount0In = balance0 > _reserve0 - amou
        uint amount1In = balance1 > _reserve1 - amou
        require(amount0In > 0 || amount1In > 0, 'Uni
        { // scope for reserve{0,1}Adjusted, avoids
```

```
        }

        _update(balance0, balance1, _reserve0, _rese
        emit Swap(msg.sender, amount0In, amount1In,
    }

    // force balances to match reserves
    function skim(address to) external lock {
        address _token0 = token0; // gas savings
        address _token1 = token1; // gas savings
        _safeTransfer(_token0, to, IERC20(_token0).b
        _safeTransfer(_token1, to, IERC20(_token1).b
    }

    // force reserves to match balances
    function sync() external lock {
        _update(IERC20(token0).balanceOf(address(thi
    }
}

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;

/**
 * @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
```

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

// File: contracts\open-zeppelin-contracts\token\ERC

pragma solidity ^0.5.0;



/**
 * @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 `ERC20Mintable`.
 *
 * *For a detailed writeup see our guide [How to imp
 * mechanisms](https://forum.zeppelin.solutions/t/ho
 *
 * 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
```

```
using SafeMath for uint256;

mapping (address => uint256) private _balances;

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

uint256 private _totalSupply;

/**
 * @dev See `IERC20.totalSupply`.
 */
function totalSupply() public view returns (uint
    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 value)
    _approve(msg.sender, spender, value);
    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 `v
 * - the caller must have allowance for `sender`
 * `amount`.
```

```
        _approve(sender, msg.sender, _allowances[sen
        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(msg.sender, spender, _allowances[ms
        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(msg.sender, spender, _allowances[ms
        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

        _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.
```

```
     * Requirements
     *
     * - `to` cannot be the zero address.
     */
    function _mint(address account, uint256 amount)
        require(account != address(0), "ERC20: mint

        _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 value) i
        require(account != address(0), "ERC20: burn

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

```
     * This is internal function is equivalent to `a
     * 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] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Destoys `amount` tokens from `account`.`
     * from the caller's allowance.
     *
     * See `_burn` and `_approve`.
     */
    function _burnFrom(address account, uint256 amou
        _burn(account, amount);
        _approve(account, msg.sender, _allowances[ac
    }
}

// File: contracts\ERC20\TokenMintERC20Token.sol
```

```
/**
 * @title TokenMintERC20Token
 * @author TokenMint (visit https://tokenmint.io)
 *
 * @dev Standard ERC20 token with burning and option
 * For full specification of ERC-20 standard see:
 * https://github.com/ethereum/EIPs/blob/master/EIPS
 */
contract TokenMintERC20Token is ERC20 {

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

    /**
     * @dev Constructor.
     * @param name name of the token
     * @param symbol symbol of the token, 3-4 chars
     * @param decimals number of decimal places of o
     * @param totalSupply total supply of tokens in
     * @param tokenOwnerAddress address that gets 10
     */
    constructor(string memory name, string memory sy
      _name = name;
      _symbol = symbol;
      _decimals = decimals;

      // set tokenOwnerAddress as owner of all token
      _mint(tokenOwnerAddress, totalSupply);
```

```
/**
 * @dev Burns a specific amount of tokens.
 * @param value The amount of lowest token units
 */
function burn(uint256 value) public {
  _burn(msg.sender, value);
}

// optional functions from ERC20 stardard

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

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

/**
 * @return the number of decimals of the token.
 */
function decimals() public view returns (uint8)
  return _decimals;
}
```

```
contract Context {

    constructor () internal { }

    function _msgSender() internal view returns (add
        return msg.sender;
    }

    function _msgData() internal view returns (bytes
        this;
        return msg.data;
    }
}

contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previ

    constructor () internal {
        _owner = _msgSender();
        emit OwnershipTransferred(address(0), _owner
    }

    function owner() public view returns (address) {
        return _owner;
    }


    modifier onlyOwner() {
```

```
    function isOwner() public view returns (bool) {
        return _msgSender() == _owner;
    }

    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0)
        _owner = address(0);
    }

    function transferOwnership(address newOwner) pub
        _transferOwnership(newOwner);
    }


    function _transferOwnership(address newOwner) in
        require(newOwner != address(0), "Ownable: ne
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

library Address {

    function isContract(address account) internal vi
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e

        assembly { codehash := extcodehash(account)
        return (codehash != 0x0 && codehash != accou
    }
```

```
    }


    function sendValue(address payable recipient, ui
        require(address(this).balance >= amount, "Ad


        (bool success, ) = recipient.call.value(amou
        require(success, "Address: unable to send va
    }
}

library SafeERC20 {
    using SafeMath for uint256;
    using Address for address;

    function safeTransfer(IERC20 token, address to,
        callOptionalReturn(token, abi.encodeWithSele
    }

    function safeTransferFrom(IERC20 token, address
        callOptionalReturn(token, abi.encodeWithSele
    }

    function safeApprove(IERC20 token, address spend
        require((value == 0) || (token.allowance(add
            "SafeERC20: approve from non-zero to non
        );
        callOptionalReturn(token, abi.encodeWithSele
    }
```

```
    }

  function safeDecreaseAllowance(
    IERC20 token,
    address spender,
    uint256 value
  )
    internal
  {
    uint256 newAllowance = token.allowance(address(t
    require(token.approve(spender, newAllowance));
}

    function callOptionalReturn(IERC20 token, bytes
        require(address(token).isContract(), "SafeER


        (bool success, bytes memory returndata) = ad
        require(success, "SafeERC20: low-level call

        if (returndata.length > 0) { // Return data

            require(abi.decode(returndata, (bool)),
        }
    }
}

contract IRewardDistributionRecipient is Ownable {
    address public rewardDistribution;

    function notifyRewardAmount(uint256 reward) exte
```

```
        _;
    }

    function setRewardDistribution(address _rewardDi
        external
        onlyOwner
    {
        rewardDistribution = _rewardDistribution;
    }
}

contract LPTokenWrapper {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    IERC20 public BUND_ETH_LP = IERC20(0xEd86244cd91

    uint256 private _totalSupply;
    mapping(address => uint256) private _balances;

    function totalSupply() public view returns (uint
        return _totalSupply;
    }

    function balanceOf(address account) public view
        return _balances[account];
    }

    function stake(uint256 amount) public {
        _totalSupply = _totalSupply.add(amount);
        _balances[msg.sender] = _balances[msg.sender
```

```
    function withdraw(uint256 amount) public {
        _totalSupply = _totalSupply.sub(amount);
        _balances[msg.sender] = _balances[msg.sender
        BUND_ETH_LP.safeTransfer(msg.sender, amount)
    }
}

contract BundEthPool is LPTokenWrapper, IRewardDistr
    IERC20 public bund = IERC20(0x8D3E855f3f55109D47
    uint256 public constant DURATION = 2419200;

    uint256 public starttime = block.timestamp;
    uint256 public periodFinish = 0;
    uint256 public rewardRate = 0;
    uint256 public lastUpdateTime;
    uint256 public rewardPerTokenStored;
    uint256 public rewardInterval = 86400;

    mapping(address => uint256) public userRewardPer
    mapping(address => uint256) public rewards;
    mapping(address => uint256) public lastTimeRewar


    event RewardAdded(uint256 reward);
    event Staked(address indexed user, uint256 amoun
    event Withdrawn(address indexed user, uint256 am
    event RewardPaid(address indexed user, uint256 r

    modifier checkStart(){
        require(block.timestamp >= starttime,"BUND-E
        _;
```

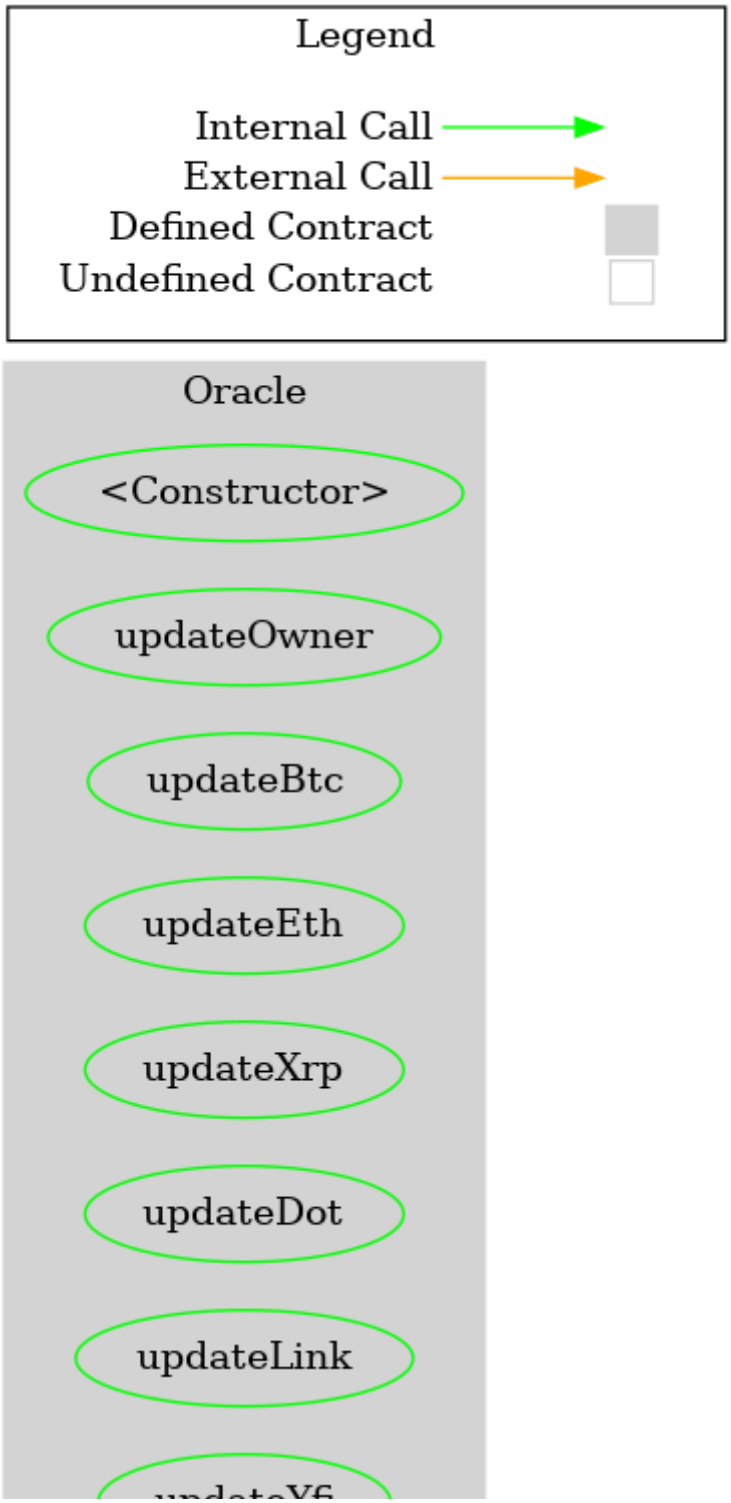
```
        rewardPerTokenStored = rewardPerToken();
        lastUpdateTime = lastTimeRewardApplicable();
        if (account != address(0)) {
            rewards[account] = earned(account);
            userRewardPerTokenPaid[account] = reward
        }
        _;
    }

    function lastTimeRewardApplicable() public view
        return Math.min(block.timestamp, periodFinis
    }

    function rewardPerToken() public view returns (u
        if (totalSupply() == 0) {
            return rewardPerTokenStored;
        }
        return
            rewardPerTokenStored.add(
                lastTimeRewardApplicable()
                    .sub(lastUpdateTime)
                    .mul(rewardRate)
                    .mul(1e18)
                    .div(totalSupply())
            );
    }

    function earned(address account) public view ret
        return
            balanceOf(account)
                .mul(rewardPerToken().sub(userReward
```

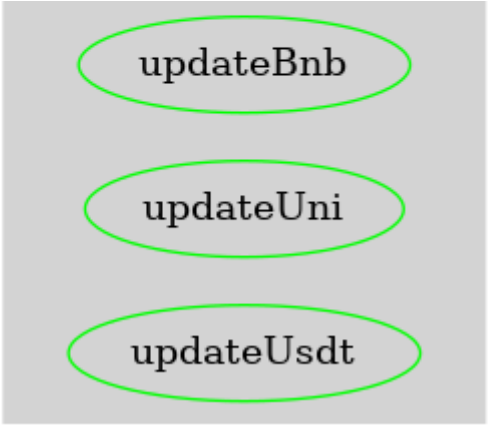
```
function collectRewardAmount() public onlyOwner
        bund.safeTransfer(msg.sender, bund.balan
}

function stake(uint256 amount) public updateRewa
    require(amount > 0, "Cannot stake 0");
    super.stake(amount);
    emit Staked(msg.sender, amount);
}

function withdraw(uint256 amount) public updateR
    require(amount > 0, "Cannot withdraw 0");
    uint256 unstakeDuration =  starttime.add(DUR
    require(block.timestamp >= unstakeDuration ,

    super.withdraw(amount);
    emit Withdrawn(msg.sender, amount);
}

function exit() external {
    withdraw(balanceOf(msg.sender));
    getReward();
}

function getReward() public updateReward(msg.sen
    uint256 reward = earned(msg.sender);

    uint256 leftTimeReward = block.timestamp.sub
    require(leftTimeReward >= rewardInterval , "
```

```
            bund.safeTransfer(msg.sender, trueReward
            lastTimeRewarded[msg.sender] = block.tim
            emit RewardPaid(msg.sender, trueReward);
        }
    }

    function notifyRewardAmount(uint256 reward)
        external
        onlyRewardDistribution
        updateReward(address(0))
    {
        if (block.timestamp > starttime) {
          if (block.timestamp >= periodFinish) {
              rewardRate = reward.div(DURATION);
          } else {
              uint256 remaining = periodFinish.sub(b
              uint256 leftover = remaining.mul(rewar
              rewardRate = reward.add(leftover).div(
          }
          lastUpdateTime = block.timestamp;
          periodFinish = block.timestamp.add(DURATIO
          emit RewardAdded(reward);
        } else {
          rewardRate = reward.div(DURATION);
          lastUpdateTime = starttime;
          periodFinish = starttime.add(DURATION);
          emit RewardAdded(reward);
        }
    }
}
```

DETAILS: ORACLE

FUNCTION GRAPH





FUNCTIONS OVERVIEW

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

Int = Internal
Ext = External
Pub = Public

+  Oracle
   - [Pub]  #
   - [Pub] updateOwner #
   - [Pub] updateBtc #
   - [Pub] updateEth #
   - [Pub] updateXrp #
   - [Pub] updateDot #
   - [Pub] updateLink #
   - [Pub] updateYfi #
```

```
    - [Pub] updateUsdt #
```

SOURCE CODE

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

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

// SPDX-License-Identifier: UNLICENSED

pragma solidity <=0.7.4;

contract Oracle{

    uint256 public BTC;
    uint256 public ETH;
    uint256 public DOT;
    uint256 public LINK;
    uint256 public XRP;
    uint256 public YFI;
    uint256 public CORE;
    uint256 public BNB;
    uint256 public UNI;
    uint256 public USDT;
```

```
    constructor(){
        owner = msg.sender;
    }

    function updateOwner(address new_owner) public {
        require(msg.sender == owner);
        owner = new_owner;
    }

    function updateBtc(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        BTC = price;
    }

    function updateEth(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        ETH = price;
    }

    function updateXrp(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        XRP = price;
    }

    function updateDot(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        DOT = price;
    }

    function updateLink(uint256 price) public {
```

```
    function updateYfi(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        YFI = price;
    }

    function updateCore(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        CORE = price;
    }

    function updateBnb(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        BNB = price;
    }

    function updateUni(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        UNI = price;
    }

    function updateUsdt(uint256 price) public {
        require(msg.sender==owner,'Cannot do this');
        USDT = price;
    }

}
```

GO HOME