SOLIDITY.*FINANCE*

Bundles Finance (BUND) - Smart Contract Audit Report

SUMMARY

Bundles

CRITICAL ISSUE ALERT - December 4th, 2020

The $BUND Bundles.Finance team has introduced probable malicious code into their project after our audit was completed. This is a developing situation; updates are available here. We are advising anyone invested in the project to unstake their tokens from the platform until this matter is resolved. At the time of writing this a very small amount of value remains in the vulnerable contracts, and no malicious actions have been taken or funds lost.

and deployment details will be available at https://solidity.finance/audits/BUNDv2/

This page is now considered deprecated.

View the updated report at the link above.

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.

> ***Audit Findings:***
>
> - *Summary: No issues from outside attackers were identified. Ensure trust in the project team.*
> - *Date: November 20th, 2020.*
>
> - *The BUND token contract is secure and cannot be minted after deployment.*

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 team and their innovative solution (to be deployed shortly) limits the risk of a compromised key to only 4% of user's funds. 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.*

We ran over 400,000 transactions interacting with this suite of contracts on a test blockchain to determine these results.

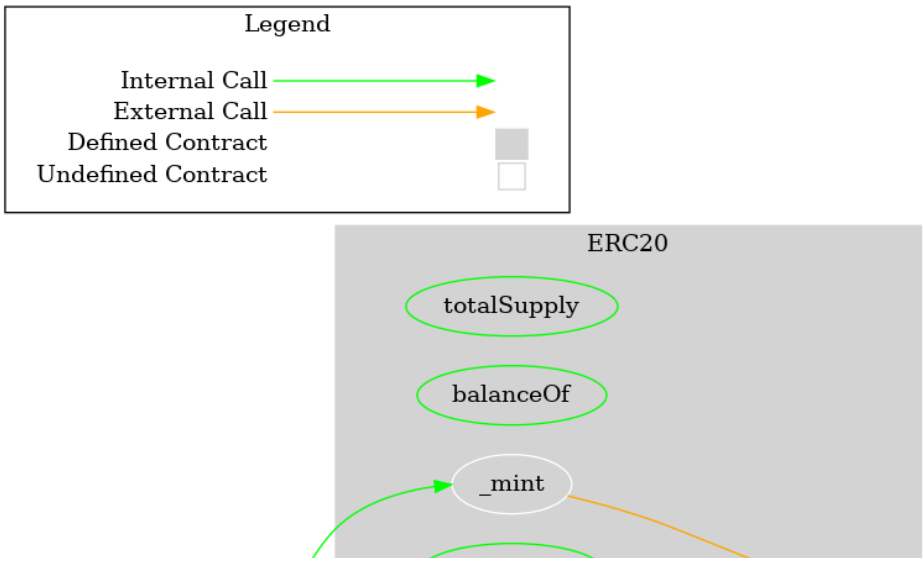
Date: November 17th, 2020

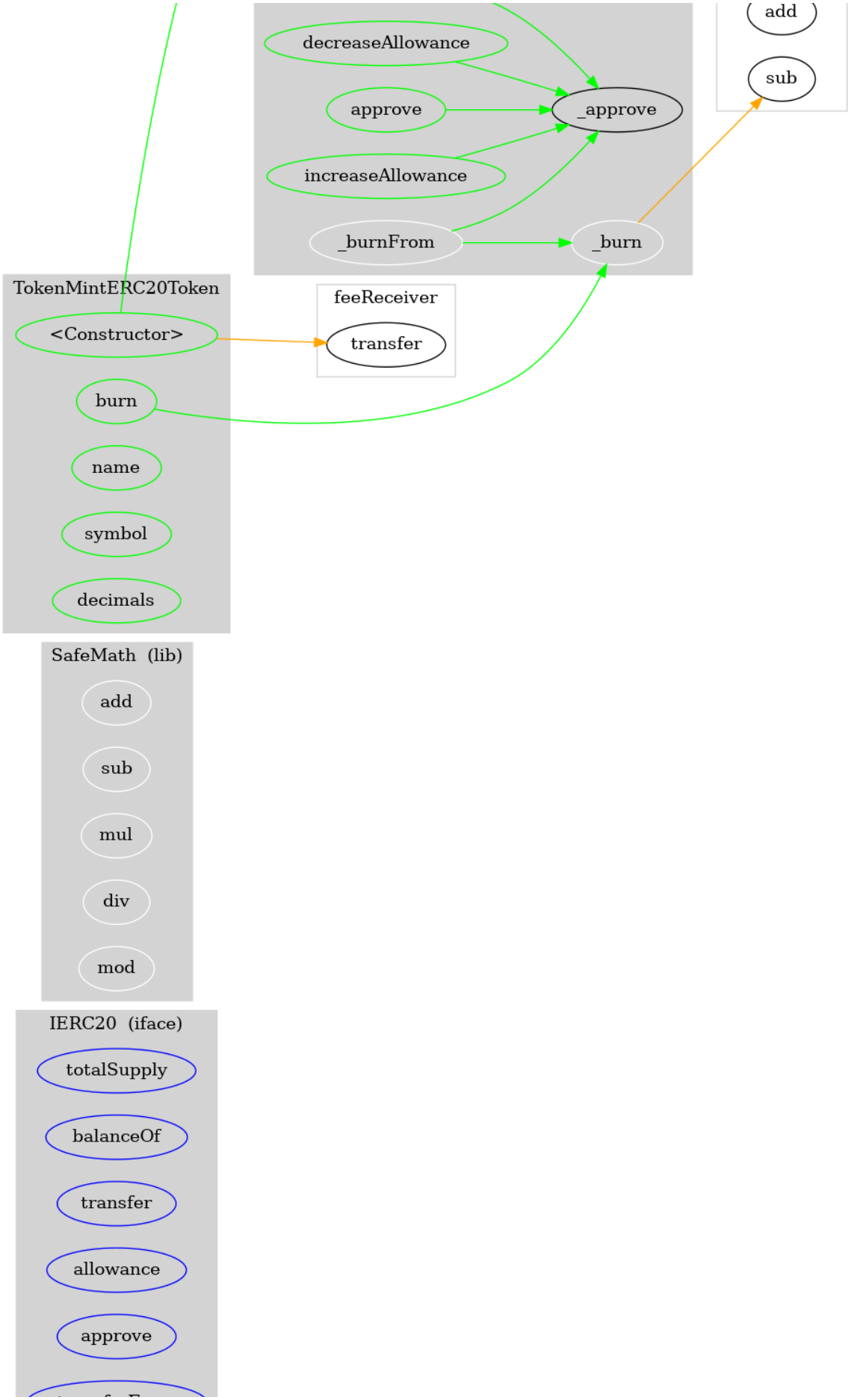
Vulnerability Category	***Notes***	***Result***
Arbitrary Storage Write	*N/A*	*PASS*
Arbitrary Jump	*N/A*	*PASS*
Delegate Call to Untrusted Contract	*N/A*	*PASS*
Dependence on Predictable Variables	*N/A*	*PASS*
Deprecated Opcodes	*N/A*	*PASS*

Vulnerability Category	Notes	Result
Ether/Token Thief	*The owner of the prediciton contract determines and sets the rewards for each user. If the owner key was compromised, only 4% of each user's staked funds would be at risk due to the implemented mitigation.*	*Warning*

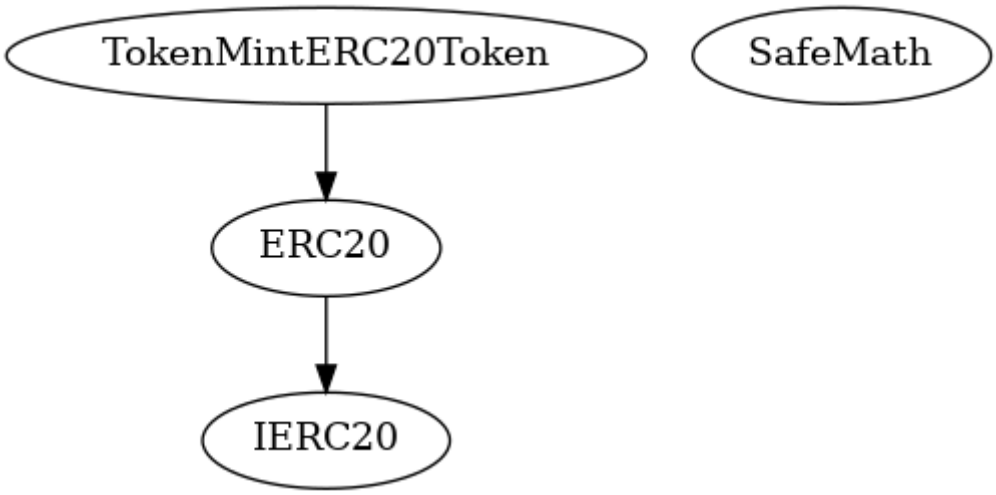
DETAILS: BUNDTOKEN

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 #
```

```
    - [Int] add
    - [Int] sub
    - [Int] mul
    - [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
```

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

```
/**
 *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
     *
```

```
     */
    function transfer(address recipient, uint256 amo

    /**
     * @dev Returns the remaining number of tokens t
     * allowed to spend on behalf of `owner` through
     * zero by default.
     *
     * This value changes when `approve` or `transfe
     */
    function allowance(address owner, address spende

    /**
     * @dev Sets `amount` as the allowance of `spend
     *
     * Returns a boolean value indicating whether th
     *
     * > 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
```

```
     *
     * Emits a `Transfer` event.
     */
    function transferFrom(address sender, address re

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

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

// 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
```

```
 * Using this library instead of the unchecked opera
 * class of bugs, so it's recommended to use it alwa
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned int
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow

        return c;
    }

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

```
    }

    /**
     * @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
```

```
     * - 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
 *
 * 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;
```

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

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

```
     *
     * 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`.
     */
    function transferFrom(address sender, address re
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sen
        return true;
    }

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

```
     * 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;
    }

    /**
```

```
     * 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.
     *
     * 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
```

```
    }

     /**
     * @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:
     *
```

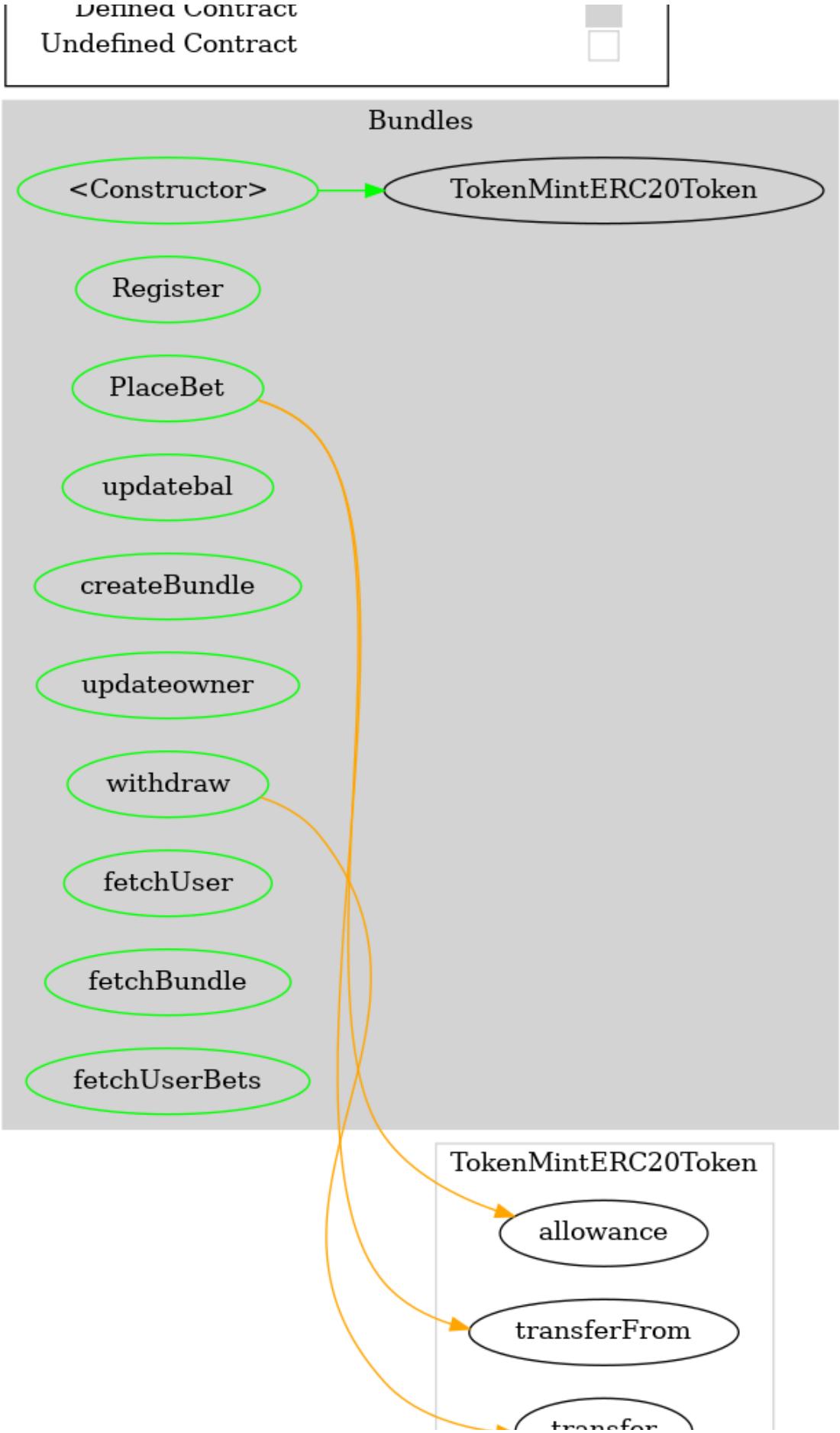
```
    function _approve(address owner, address spender
        require(owner != address(0), "ERC20: approve
        require(spender != address(0), "ERC20: appro

        _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

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:
```

```
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);

  // 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);
```

```
    /**
     * @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;
    }
}
```

DETAILS: BUNDLES

FUNCTION GRAPH



FUNCTIONS OVERVIEW

```
+  Bundles
  - [Pub]  #
  - [Pub] Register #
  - [Pub] PlaceBet #
  - [Pub] updatebal #
  - [Pub] createBundle #
  - [Pub] updateowner #
  - [Pub] withdraw #
  - [Pub] fetchUser
  - [Pub] fetchBundle
  - [Pub] fetchUserBets
```

SOURCE CODE

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

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

pragma solidity <=0.7.5;
```

```
uint256 public bundleId = 1;
address public owner;
TokenMintERC20Token public bundle_address;

uint256 lastcreated;

struct UserBets{
    uint256[10] bundles;
    bool betted;
}

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

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

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

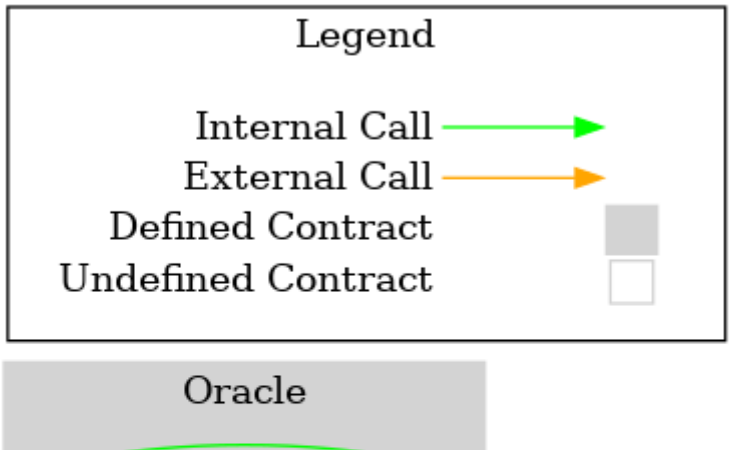
```
    }

    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;
    }

    function PlaceBet(uint256[10] memory _bundle,uin
        require(_bundleId <= bundleId,'Invalid Bundl
        require(bundle_address.allowance(msg.sender,
        Bundle storage b = bundle[_bundleId];
        require(b.endtime >= block.timestamp,'Ended'
        User storage us = user[msg.sender];
        require(us.active == true,'Register to parti
        UserBets storage u = bets[msg.sender][_bundl
        require(u.betted == false, 'Already Voted');
        us.bundles.push(_bundleId);
        us.balance = us.balance+_amount;
        u.betted = true;
        u.bundles = _bundle;
        bundle_address.transferFrom(msg.sender,addre
        return true;
    }


    function updatebal(address _user,uint256 _newbal
        require(msg.sender == owner,'Not Owner');
                require(_reward <=40,'Invalid Reward
```

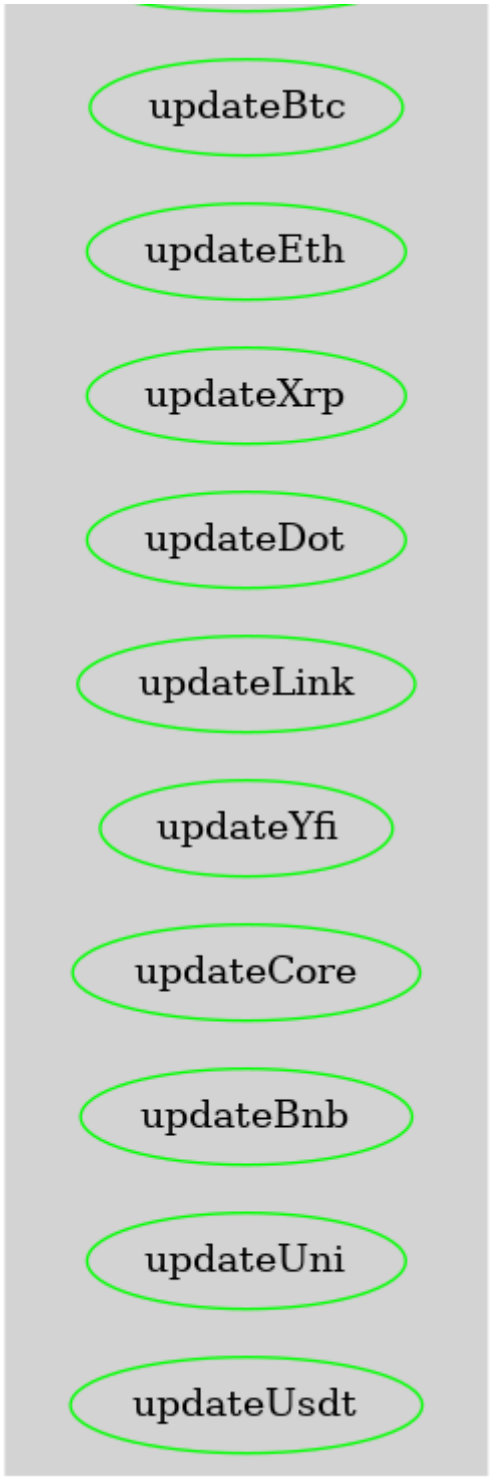
```
        us.balance = 0;
        return true;
    }

    function createBundle(uint256[10] memory _prices
        require(msg.sender == owner,'Not Owner');
        require( block.timestamp > lastcreated + 7 d
        Bundle storage b = bundle[bundleId];
        b.prices = _prices;
        b.startime = block.timestamp;
        lastcreated = block.timestamp;
        b.endtime = block.timestamp + 7 days;
        b.stakingends = block.timestamp + 1 days;
        bundleId = bundleId + 1;
        return true;
    }

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

    function withdraw() public returns(bool){
       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;
    }
```

```
        return(us.bundles,us.username,us.balance,us.
    }

    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.betted);
    }

}
```

DETAILS: ORACLE

FUNCTION GRAPH





FUNCTIONS OVERVIEW

```
Int = Internal
Ext = External
Pub = Public

+  Oracle
   - [Pub]  #
   - [Pub] updateOwner #
   - [Pub] updateBtc #
   - [Pub] updateEth #
   - [Pub] updateXrp #
   - [Pub] updateDot #
   - [Pub] updateLink #
   - [Pub] updateYfi #
   - [Pub] updateCore #
   - [Pub] updateBnb #
   - [Pub] updateUni #
   - [Pub] updateUsdt #
```

SOURCE CODE

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

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

```
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;


    address public owner;

    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;
    }
```

```
        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 {
        require(msg.sender==owner,'Cannot do this');
        LINK = price;
    }

    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;
```

```
        require(msg.sender==owner,'Cannot do this');
        UNI = price;
    }

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

}
```

PRINT EXPANDED SECTIONS

GO HOME