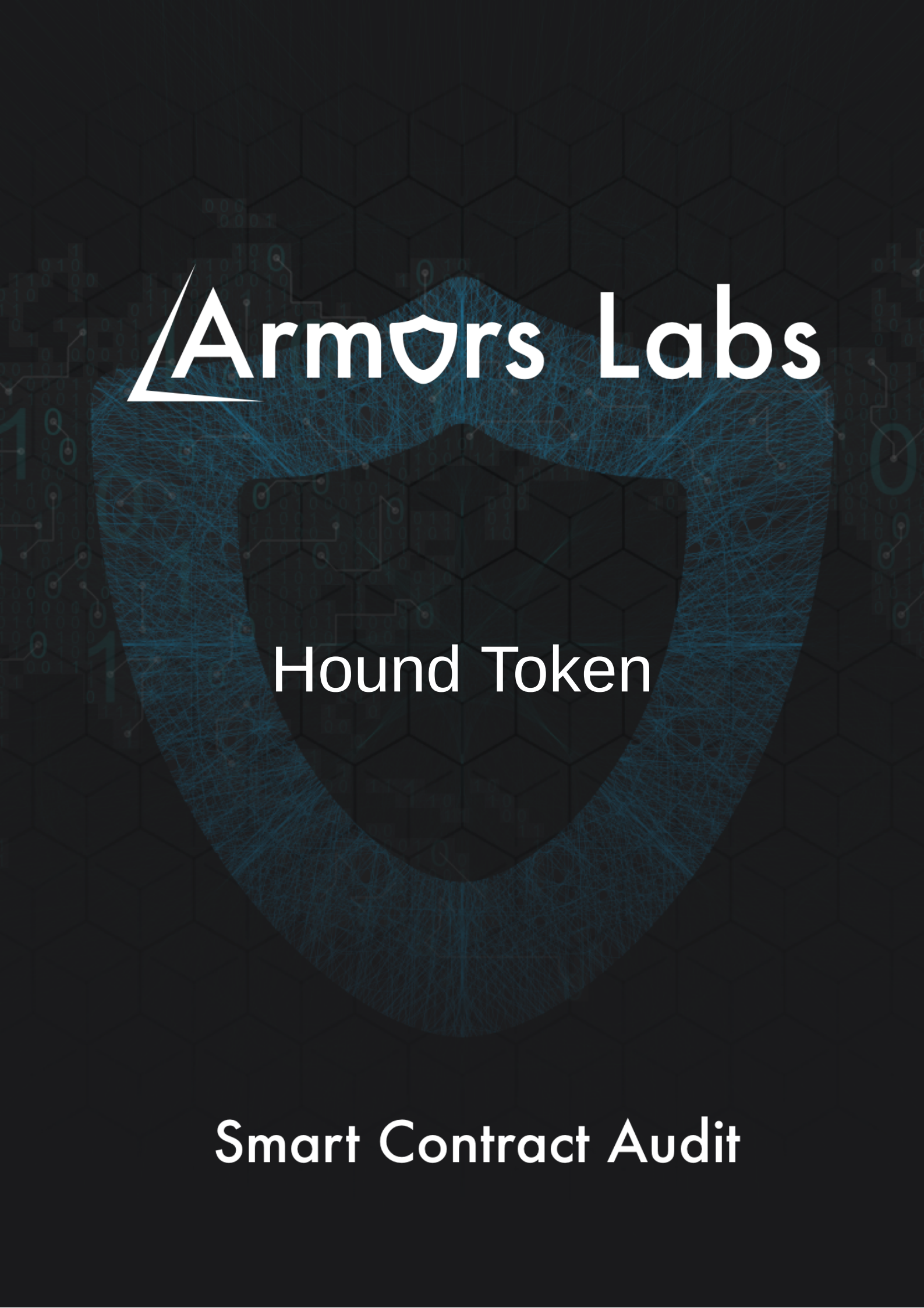

Armors Labs
Hound Token
Smart Contract Audit

- Hound Token 审核总结
- Hound Token 审计
  - 文件信息
    - 审计结果
    - 免责声明
    - 审核的目标文件
  - 漏洞分析
    - 漏洞分布
    - 合约文件
    - 审计结果分析
      - 重入
      - 算法上下溢出
      - 货币异常
      - Delegatecall
      - 默认可见性
      - 随机数误区
      - 外部合约引用
      - 未解决的待办事项
      - 短地址/参数攻击
      - 未检查的CALL返回值
      - 条件竞争/优先提交
      - 拒绝服务（DOS）
      - 锁定时间戳操作
      - 构造函数失控
      - 未初始化的存储指针
      - 浮点数和数值精度
      - tx.origin 身份验证
      - 权限限制

Hound Token 审核总结

项目名称：Hound Token 合约

项目地址：无

代码地址: https://bscscan.com/address/0x1e4402fa427a7a835fc64ea6d051404ce767a569#code

代码版本: 无

项目目标: Hound Token 合约审核

区块链: Binance Smart Chain（BSC）

审计结果: 通过

审计信息

审计编号: 0X202203150026

审计小组: Armors Labs

审计校对: https://armors.io/#project-cases

Hound Token 审计

Hound Token 团队要求我们审查和审计他们的 Hound Token 合约。我们查看了代码，现在公布了结果。

以下是我们的评估和建议，按重要性排序。

文件信息

项目名称	审计人员	版本	日期
Hound Token Audit	Rock, Sophia, Rushairer, Rico, David, Alice	1.0.0	2022-03-15

审计结果

请注意，截至发布之日，上述审查反映了对已知安全模式的当前理解，因为它们与 Hound Token 合约有关。以上内容不应理解为投资建议。

基于当前基础区块链和智能合约的广泛认可的安全状态，本审计报告自输出之日起3个月内有效。

免责声明

Armors实验室的报告不应也不应被视为对任何特定项目或团队的“批准”或“不批准”。这些报告不是也不应该被视为任何团队创造的任何“产品”或“资产”的经济或价值指标。Armors不包括测试或审计与外部合同或服务（如Unicrypt、Uniswap、PancakeSwap等）的集成

Armors实验室的报告代表了一个广泛的审计过程，旨在帮助我们的客户提高代码质量，同时降低加密令牌和区块链技术带来的高风险。Armors不保证同意分析的技术的安全性或功能性。

Armors实验室假定提供的信息没有丢失、篡改、删除或隐藏。如果提供的信息丢失、篡改、删除、隐藏或以不符合实际情况的方式反映，Armors Labs不对由此造成的损失和不良影响负责。

Armors实验室的审计不应以任何方式用于就投资或参与任何特定项目做出决策。这些报告绝不提供投资建议，也不应被用作任何类型的投资建议。

审核的目标文件

file	md5
Hound Token.sol	5dacde6bc9148c0ad28ede60dba1a410

漏洞分析

漏洞分布

漏洞等级	数量
严重程度	0
严重程度高	0
中等严重程度	0
低严重性	0

###审计结果总结

漏洞	状态
重入	安全
算法上下溢出	安全
货币异常	安全
Delegatecall	安全
默认可见性	安全
随机数误区	安全
外部合约引用	安全
短地址/参数攻击	安全
未检查的CALL返回值	安全
条件竞争/优先提交	安全
拒绝服务（DOS）	安全
锁定时间戳操作	安全
构造函数失控	安全
未初始化的存储指针	安全

漏洞	状态
浮点数和数值精度	安全
tx.origin 身份验证	安全
权限限制	安全

合约文件

```
/**
 *Submitted for verification at BscScan.com on 2022-03-09
*/

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

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

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

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

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

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

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

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

    function _msgData() internal view virtual returns (bytes calldata) {
        this; // silence state mutability warning without generating bytecode - see https://github.co
        return msg.data;
    }
}

interface IUniswapV2Router01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
    function removeLiquidity(
        address tokenA,
        address tokenB,
```

```
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETH(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountToken, uint amountETH);
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountToken, uint amountETH);
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapExactETHForTokens(uint amountOutMin, address[] calldata path, address to, uint deadl
    external
    payable
    returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address
    external
    returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path, address
    external
    returns (uint[] memory amounts);
    function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline
    external
    payable
    returns (uint[] memory amounts);

    function quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) external pure returns (uint
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) external pure returns (uint
    function getAmountsOut(uint amountIn, address[] calldata path) external view returns (uint[] memo
```

```
    function getAmountsIn(uint amountOut, address[] calldata path) external view returns (uint[] memo
}

interface IUniswapV2Router02 is IUniswapV2Router01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountETH);

    function swapExactTokensForTokensSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
    function swapExactETHForTokensSupportingFeeOnTransferTokens(
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external payable;
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
}

interface IUniswapV2Factory {
    event PairCreated(address indexed token0, address indexed token1, address pair, uint);

    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);

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

    function createPair(address tokenA, address tokenB) external returns (address pair);

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

interface IUniswapV2Pair {
    event Approval(address indexed owner, address indexed spender, uint value);
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external pure returns (string memory);
    function symbol() external pure returns (string memory);
```

```
    function decimals() external pure returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view returns (uint);

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external returns (bool);

    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint);

    function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, by

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    function MINIMUM_LIQUIDITY() external pure returns (uint);
    function factory() external view returns (address);
    function token0() external view returns (address);
    function token1() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112 reserve1, uint32 blockTim
    function price0CumulativeLast() external view returns (uint);
    function price1CumulativeLast() external view returns (uint);
    function kLast() external view returns (uint);


    function burn(address to) external returns (uint amount0, uint amount1);
    function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) external;
    function skim(address to) external;
    function sync() external;

    function initialize(address, address) external;
}

interface IERC20Metadata is IERC20 {
    /**
     * @dev Returns the name of the token.
     */
    function name() external view returns (string memory);

    /**
     * @dev Returns the symbol of the token.
     */
    function symbol() external view returns (string memory);

    /**
     * @dev Returns the decimals places of the token.
     */
    function decimals() external view returns (uint8);
}

contract Ownable is Context {
    address private _owner;

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

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

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

    modifier onlyOwner() {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
        _;
    }

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

    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
        if (a == 0) {
            return 0;
        }

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

        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
```

```
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}

contract ERC20 is Context, IERC20, IERC20Metadata {
    using SafeMath for uint256;

    mapping(address => uint256) private _balances;

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

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    /**
     * @dev Sets the values for {name} and {symbol}.
     *
     * The default value of {decimals} is 18. To select a different value for
     * {decimals} you should overload it.
     *
     * All two of these values are immutable: they can only be set once during
     * construction.
     */
    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }

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

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

    /**
     * @dev Returns the number of decimals used to get its user representation.
     * For example, if `decimals` equals `2`, a balance of `505` tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
     *
     * Tokens usually opt for a value of 18, imitating the relationship between
     * Ether and Wei. This is the value {ERC20} uses, unless this function is
     * overridden;
```

```
     *
     * NOTE: This information is only used for _display_ purposes: it in
     * no way affects any of the arithmetic of the contract, including
     * {IERC20-balanceOf} and {IERC20-transfer}.
     */
    function decimals() public view virtual override returns (uint8) {
        return 18;
    }

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

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

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

    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spender) public view virtual override returns (uint256)
        return _allowances[owner][spender];
    }

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

    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance. This is not
     * required by the EIP. See the note at the beginning of {ERC20}.
     *
     * Requirements:
     *
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     * - the caller must have allowance for ``sender``'s tokens of at least
```

```
     * `amount`.
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer
        return true;
    }

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

    /**
     * @dev Atomically decreases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC2
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `recipient`.
     *
     * This is internal function is equivalent to {transfer}, and can be used to
     * e.g. implement automatic token fees, slashing mechanisms, etc.
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     */
    function _transfer(
        address sender,
        address recipient,
        uint256 amount
```

```
    ) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

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

    /** @dev Creates `amount` tokens and assigns them to `account`, increasing
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to the zero address.
     *
     * Requirements:
     *
     * - `account` cannot be the zero address.
     */
    function _cast(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");

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

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

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

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

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

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

```
        address spender,
        uint256 amount
    ) internal virtual {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

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

    /**
     * @dev Hook that is called before any transfer of tokens. This includes
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will be minted for `to`.
     * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks]
     */
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
}

contract TokenDividendTracker is Ownable {
    using SafeMath for uint256;

    address[] public shareholders;
    uint256 public currentIndex;
    mapping(address => bool) private _updated;
    mapping (address => uint256) public shareholderIndexes;

    address public  uniswapV2Pair;
    address public lpRewardToken;
    // 上次分红时间
    uint256 public LPRewardLastSendTime;

    constructor(address uniswapV2Pair_, address lpRewardToken_){
        uniswapV2Pair = uniswapV2Pair_;
        lpRewardToken = lpRewardToken_;
    }

    function resetLPRewardLastSendTime() public onlyOwner {
        LPRewardLastSendTime = 0;
    }

    // LP分红发放
    function process(uint256 gas) external onlyOwner {
        uint256 shareholderCount = shareholders.length;

        if(shareholderCount == 0) return;
        uint256 nowbanance = IERC20(lpRewardToken).balanceOf(address(this));

        uint256 gasUsed = 0;
        uint256 gasLeft = gasleft();

        uint256 iterations = 0;

        while(gasUsed < gas && iterations < shareholderCount) {
```

```
            if(currentIndex >= shareholderCount){
                currentIndex = 0;
                LPRewardLastSendTime = block.timestamp;
                return;
            }

            uint256 amount = nowbanance.mul(IERC20(uniswapV2Pair).balanceOf(shareholders[currentIndex
            if( amount == 0) {
                currentIndex++;
                iterations++;
                return;
            }
            if(IERC20(lpRewardToken).balanceOf(address(this))  < amount ) return;
            IERC20(lpRewardToken).transfer(shareholders[currentIndex], amount);
            gasUsed = gasUsed.add(gasLeft.sub(gasleft()));
            gasLeft = gasleft();
            currentIndex++;
            iterations++;
        }
    }
    // 根据条件自动将交易账户加入、退出流动性分红
    function setShare(address shareholder) external onlyOwner {
        if(_updated[shareholder] ){
            if(IERC20(uniswapV2Pair).balanceOf(shareholder) == 0) quitShare(shareholder);
            return;
        }
        if(IERC20(uniswapV2Pair).balanceOf(shareholder) == 0) return;
        addShareholder(shareholder);
        _updated[shareholder] = true;

    }
    function quitShare(address shareholder) internal {
        removeShareholder(shareholder);
        _updated[shareholder] = false;
    }

    function addShareholder(address shareholder) internal {
        shareholderIndexes[shareholder] = shareholders.length;
        shareholders.push(shareholder);
    }

    function removeShareholder(address shareholder) internal {
        shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-1];
        shareholderIndexes[shareholders[shareholders.length-1]] = shareholderIndexes[shareholder];
        shareholders.pop();
    }

}


contract Hound is ERC20, Ownable {
    using SafeMath for uint256;

    IUniswapV2Router02 public uniswapV2Router;
    address public  uniswapV2Pair;
    bool private swapping;

    uint256 public swapTokensAtAmount;

    uint256 public deadFee = 3;
    uint256 public liquidityFee = 4;
    uint256 public marketingFee = 1;
    uint256 public foundationFee = 1;
    uint256 public lpRewardFee = 5;
    address public lpRewardToken = 0x55d398326f99059fF775485246999027B3197955;
```

```
    uint256 public AmountLiquidityFee;
    uint256 public AmountLpRewardFee;

    address public marketingWalletAddress;
    address public foundationWalletAddress;
    address public liquidityReceiveAddress;
    address public deadWallet = 0x000000000000000000000000000000000000dEaD;


    mapping (address => bool) private _isExcludedFromFees;

    TokenDividendTracker public dividendTracker;

    address private fromAddress;
    address private toAddress;
    mapping (address => bool) isDividendExempt;


    uint256 public minPeriod = 86400;

    uint256 distributorGas = 200000;

    event ExcludeFromFees(address indexed account, bool isExcluded);
    event ExcludeMultipleAccountsFromFees(address[] accounts, bool isExcluded);

    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );


    constructor(
        string memory name_,
        string memory symbol_,
        uint256 totalSupply_,
        address marketingWalletAddr_,
        address foundationWalletAddress_,
        address liquidityReceiveAddress_
    ) payable ERC20(name_, symbol_) {
        uint256 totalSupply = totalSupply_ * (10**18);
        swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%;

        IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E2
        address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

        uniswapV2Router = _uniswapV2Router;
        uniswapV2Pair = _uniswapV2Pair;

        marketingWalletAddress = marketingWalletAddr_;
        foundationWalletAddress = foundationWalletAddress_;
        liquidityReceiveAddress = liquidityReceiveAddress_;
        dividendTracker = new TokenDividendTracker(uniswapV2Pair, lpRewardToken);


        excludeFromFees(owner(), true);
        excludeFromFees(marketingWalletAddress, true);
        excludeFromFees(foundationWalletAddress, true);
        excludeFromFees(address(this), true);
        excludeFromFees(address(dividendTracker), true);

        isDividendExempt[address(this)] = true;
        isDividendExempt[address(0)] = true;
        isDividendExempt[address(dividendTracker)] = true;
```

```
        _cast(owner(), totalSupply);
    }

    receive() external payable {}


    function excludeFromFees(address account, bool excluded) public onlyOwner {
        if(_isExcludedFromFees[account] != excluded){
            _isExcludedFromFees[account] = excluded;
            emit ExcludeFromFees(account, excluded);
        }
    }


    function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyO
        for(uint256 i = 0; i < accounts.length; i++) {
            _isExcludedFromFees[accounts[i]] = excluded;
        }
        emit ExcludeMultipleAccountsFromFees(accounts, excluded);
    }


    function setMarketingWallet(address payable wallet) external onlyOwner{
        marketingWalletAddress = wallet;
    }


    function setFoundationWallet(address addr) public onlyOwner {
        foundationWalletAddress = addr;
    }


    function isExcludedFromFees(address account) public view returns(bool) {
        return _isExcludedFromFees[account];
    }


    function setSwapTokensAtAmount(uint256 amount) public onlyOwner {
        swapTokensAtAmount = amount;
    }


    function setLiquidityFee(uint256 val) public onlyOwner {
        liquidityFee = val;
    }


    function setMarketingFee(uint256 val) public onlyOwner {
        marketingFee = val;
    }


    function setFoundationFee(uint256 val) public onlyOwner {
        foundationFee = val;
    }


    function setDeadFee(uint256 val) public onlyOwner {
        deadFee = val;
    }


    function setLpRewardFee(uint256 val) public onlyOwner {
        lpRewardFee = val;
    }
```

```
function setMinPeriod(uint256 number) public onlyOwner {
    minPeriod = number;
}


function setLiquidityReceiveAddress(address val) public onlyOwner {
    liquidityReceiveAddress = val;
}


function resetLPRewardLastSendTime() public onlyOwner {
    dividendTracker.resetLPRewardLastSendTime();
}


function updateDistributorGas(uint256 newValue) public onlyOwner {
    require(newValue >= 100000 && newValue <= 500000, "distributorGas must be between 200,000 and
    require(newValue != distributorGas, "Cannot update distributorGas to same value");
    distributorGas = newValue;
}

function _transfer(
    address from,
    address to,
    uint256 amount
) internal override {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    if(amount == 0) { super._transfer(from, to, 0); return;}

    uint256 contractTokenBalance = balanceOf(address(this));
    bool canSwap = contractTokenBalance >= swapTokensAtAmount;
    if( canSwap &&
        !swapping &&
        from != uniswapV2Pair &&
        from != owner() &&
        to != owner()
    ) {
        swapping = true;
        if(AmountLiquidityFee > 0){
            swapAndLiquify(AmountLiquidityFee);
            AmountLiquidityFee = 0;
        }
        if(AmountLpRewardFee > 0){
            swapLPRewardToken(AmountLpRewardFee);
            AmountLpRewardFee = 0;
        }
        swapping = false;
    }

    bool takeFee = !swapping;
    if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
        takeFee = false;
    }

    if(takeFee) {
        if(from != uniswapV2Pair){
            uint256 minHolderAmount = balanceOf(from).mul(90).div(100);
            if(amount > minHolderAmount){
                amount = minHolderAmount;
            }
        }
        amount =  takeAllFee(from, amount);

    }
```

```
        super._transfer(from, to, amount);

        if(fromAddress == address(0) )fromAddress = from;
        if(toAddress == address(0) )toAddress = to;
        if(!isDividendExempt[fromAddress] && fromAddress != uniswapV2Pair )   try dividendTracker.set
        if(!isDividendExempt[toAddress] && toAddress != uniswapV2Pair ) try dividendTracker.setShare(
        fromAddress = from;
        toAddress = to;

        if(  !swapping &&
        from != owner() &&
        to != owner() &&
        from !=address(this) &&
        dividendTracker.LPRewardLastSendTime().add(minPeriod) <= block.timestamp
        ) {
            try dividendTracker.process(distributorGas) {} catch {}
        }
    }


    function takeAllFee(address from,uint256 amount) private returns(uint256 amountAfter) {
        amountAfter = amount;

        uint256 DFee = amount.mul(deadFee).div(100);
        if(DFee > 0) super._transfer(from, deadWallet, DFee);
        amountAfter = amountAfter.sub(DFee);

        uint256 MFee = amount.mul(marketingFee).div(100);
        if(MFee > 0) super._transfer(from, marketingWalletAddress, MFee);
        amountAfter = amountAfter.sub(MFee);

        uint256 FFee = amount.mul(foundationFee).div(100);
        if(FFee > 0) super._transfer(from, foundationWalletAddress, FFee);
        amountAfter = amountAfter.sub(FFee);

        uint256 LFee = amount.mul(liquidityFee).div(100);
        AmountLiquidityFee += LFee;
        amountAfter = amountAfter.sub(LFee);

        uint256 LPFee = amount.mul(lpRewardFee).div(100);
        AmountLpRewardFee += LPFee;
        amountAfter = amountAfter.sub(LPFee);

        super._transfer(from, address(this), LFee.add(LPFee));
    }



    function swapAndLiquify(uint256 tokens) private {
        // split the contract balance into halves
        uint256 half = tokens.div(2);
        uint256 otherHalf = tokens.sub(half);

        uint256 initialBalance = address(this).balance;

        // swap tokens for ETH
        swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+liquify is triggered

        // how much ETH did we just swap into?
        uint256 newBalance = address(this).balance.sub(initialBalance);
        // add liquidity to uniswap
        addLiquidity(otherHalf, newBalance);
        emit SwapAndLiquify(half, newBalance, otherHalf);
    }
```

```
    function swapTokensForEth(uint256 tokenAmount) private {
        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // make the swap
        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );

    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityReceiveAddress,
            block.timestamp
        );

    }

    function swapLPRewardToken(uint256 tokenAmount) private {
        address[] memory path = new address[](3);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        path[2] = lpRewardToken;
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
            tokenAmount,
            0,
            path,
            address(dividendTracker),
            block.timestamp
        );

    }

}
```


审计结果分析

重入

- **说明:**
  智能合约的特点之一是能够调用和利用其他外部合约的代码。 合约通常还处理区块链货币，因此经常将区块链货币发送到各种外部用户地址。 调用外部合约，或者向地址发送区块链货币的操作，需要合约提交外部调用。 这些

外部调用可能会被攻击者劫持，从而迫使合约执行进一步的代码（即通过回退函数），包括对自身的回调。 因此代码执行“重入”合约。 这种攻击被用于臭名昭著的 DAO 黑客攻击。

- **检测结果:**

  通过！

- **安全建议:**
  不.

算法上下溢出

- **说明:**
  虚拟机 (EVM) 为整数指定固定大小的数据类型。 这意味着一个整数变量，只有它可以表示的一定范围的数字。 例如，一个 uint8 只能存储 [0,255] 范围内的数字。 尝试将 256 存储到 uint8 将导致 0。如果不小心，如果未检查用户输入并且执行的计算导致数字超出存储它们的数据类型范围，则可以利用 Solidity 中的变量。
- **检测结果:**

  通过！

- **安全建议:**
  不.

货币异常

- **说明:**
  通常，当区块链货币被发送到合约时，它必须执行回退函数或合约中描述的其他函数。 有两个例外，区块链货币可以存在于合约中而无需执行任何代码。 依赖于发送到合约的每个区块链货币的代码执行的合约可能容易受到区块链货币被强制发送到合约的攻击。
- **检测结果:**

  通过！

- **安全建议:** 不.

Delegatecall

- **说明:**
  CALL 和 DELEGATECALL 操作码在允许开发人员模块化他们的代码方面很有用。 对合约的标准外部消息调用由 CALL 操作码处理，代码在外部合约/函数的上下文中运行。 DELEGATECALL 操作码与标准消息调用相同，只是在目标地址执行的代码在调用合约的上下文中运行，而且 msg.sender 和 msg.value 保持不变。 此功能支持库的实现，开发人员可以借此为未来的合约创建可重用的代码。
- **检测结果:**

  通过！

- **安全建议:** 不.

默认可见性

- **说明:**
  Solidity 中的函数具有可见性说明符，它规定了如何调用函数。 可见性决定了 whBlockchain Currency 一个函数可以被用户外部调用，也可以被其他衍生合约调用，只能在内部调用，也可以只在外部调用。 有四个可见性说明符，在 Solidity Docs 中有详细描述。 函数默认为公共，允许用户在外部调用它们。 不正确使用可见性说明符可能会导致智能合约中的一些破坏性漏洞，如本节所述。
- **检测结果:**

  ```
  通过！
  ```

- **安全建议:** 不.

随机数误区

- **说明:**
  区块链上的所有交易都是确定性的状态转换操作。 这意味着每笔交易都会修改生态系统的全球状态，并且它以可计算的方式进行，没有不确定性。 这最终意味着区块链生态系统内部没有熵或随机性的来源。 Solidity 中没有 rand() 函数。 实现去中心化熵（随机性）是一个成熟的问题，并且已经提出了许多想法来解决这个问题（例如，参见 RandDAO 或使用 Vitalik 在本文中描述的哈希链）。
- **检测结果:**

  ```
  通过！
  ```

- **安全建议:** 不.

外部合约引用

- **说明:**
  全球计算机的好处之一是能够重用代码并与已经部署在网络上的合约进行交互。 结果，大量合约引用外部合约，并且在一般操作中使用外部消息调用与这些合约进行交互。 这些外部消息调用可以以一些不明显的方式掩盖恶意行为者的意图，我们将对此进行讨论。
- **检测结果:**

  ```
  通过！
  ```

- **安全建议:** 不.

未解决的待办事项

- **说明:**
  检查未解决的待办事项
- **检测结果:**

  ```
  通过！
  ```

- **安全建议:** 不.

短地址/参数攻击

- **说明:**
  这种攻击不是专门针对 Solidity 合约本身执行的，而是针对可能与其交互的第三方应用程序。 我添加这种攻击是

为了完整性，并了解如何在合约中操纵参数。

- **检测结果:**

  ```
  通过!
  ```

- **安全建议:** 不.

未检查的CALL返回值

- **说明:** 有多种方法可以可靠地执行外部调用。 向外部账户发送区块链货币通常是通过 transfer() 方法执行的。 但是，也可以使用 send() 函数，对于更通用的外部调用，可以直接在 solidity 中使用 CALL 操作码。 call() 和 send() 函数返回一个布尔值，指示调用是成功还是失败。 因此，这些函数有一个简单的警告，如果外部调用（由 call() 或 send() 初始化）失败，执行这些函数的事务将不会恢复，而 call() 或 send() 将简单地返回 false。 当未检查返回值时，会出现一个常见的陷阱，而开发人员希望会发生还原。
- **检测结果:**

  ```
  通过!
  ```

- **安全建议:** 不.

条件竞争/优先提交

- **说明:**
  对其他合约的外部调用和底层区块链的多用户特性相结合，导致了各种潜在的 Solidity 陷阱，用户竞相执行代码以获得意外状态。 重入是这种竞争条件的一个例子。 在本节中，我们将更一般地讨论区块链上可能发生的不同类型的竞争条件。 关于这个主题有很多很好的帖子，其中一些是：Wiki - 安全、DASP - Front-Running 和共识 - 智能合约最佳实践。
- **检测结果:**

  ```
  通过!
  ```

- **安全建议:** 不.

拒绝服务（DOS）

- **说明:**
  这一类别非常广泛，但基本上包括攻击，用户可以让合约在短时间内无法运行，或者在某些情况下永久无法运行。 这可能会永远将区块链货币困在这些合约中，就像 Second Parity MultiSig hack 的情况一样。
- **检测结果:**

  ```
  通过!
  ```

- **安全建议:** 不.

锁定时间戳操作

- **说明:**
  区块时间戳历来被用于各种应用，例如随机数的熵（有关更多详细信息，请参阅熵错觉部分）、在一段时间内锁定资金以及各种与时间相关的状态变化条件语句。 矿工有能力稍微调整时间戳，如果在智能合约中错误地使用区块时间戳，这可能会非常危险。

- **检测结果:**

  通过!

- **安全建议:** 不.

构造函数失控

- **说明:**
  构造函数是在初始化合约时经常执行关键的特权任务的特殊函数。 在solidity v0.4.22 之前，构造函数被定义为与包含它们的合约同名的函数。 因此，当合约名称在开发过程中发生更改时，如果构造函数名称未更改，它将成为一个正常的可调用函数。 正如你可以想象的那样，这可以（并且已经）导致一些有趣的合同黑客。
- **检测结果:**

  通过!

- **安全建议:** 不.

未初始化的存储指针

- **说明:**
  EVM 将数据存储为存储器或存储器。 在开发合约时，强烈建议准确了解这是如何完成的以及函数局部变量的默认类型。 这是因为不恰当地初始化变量可能会产生易受攻击的合约。
- **检测结果:**

  通过!

- **安全建议:** 不.

浮点数和数值精度

- **说明:**
  在撰写本文时（Solidity v0.4.24），不支持定点或浮点数。 这意味着浮点表示必须使用 Solidity 中的整数类型。 如果没有正确实施，这可能会导致错误/漏洞。
- **检测结果:**

  通过!

- **安全建议:** 不.

tx.origin 身份验证

- **说明:**
  Solidity 有一个全局变量 tx.origin，它遍历整个调用栈，返回最初发送调用（或交易）的账户地址。 在智能合约中使用此变量进行身份验证会使合约容易受到类似网络钓鱼的攻击。
- **检测结果:**

  通过!

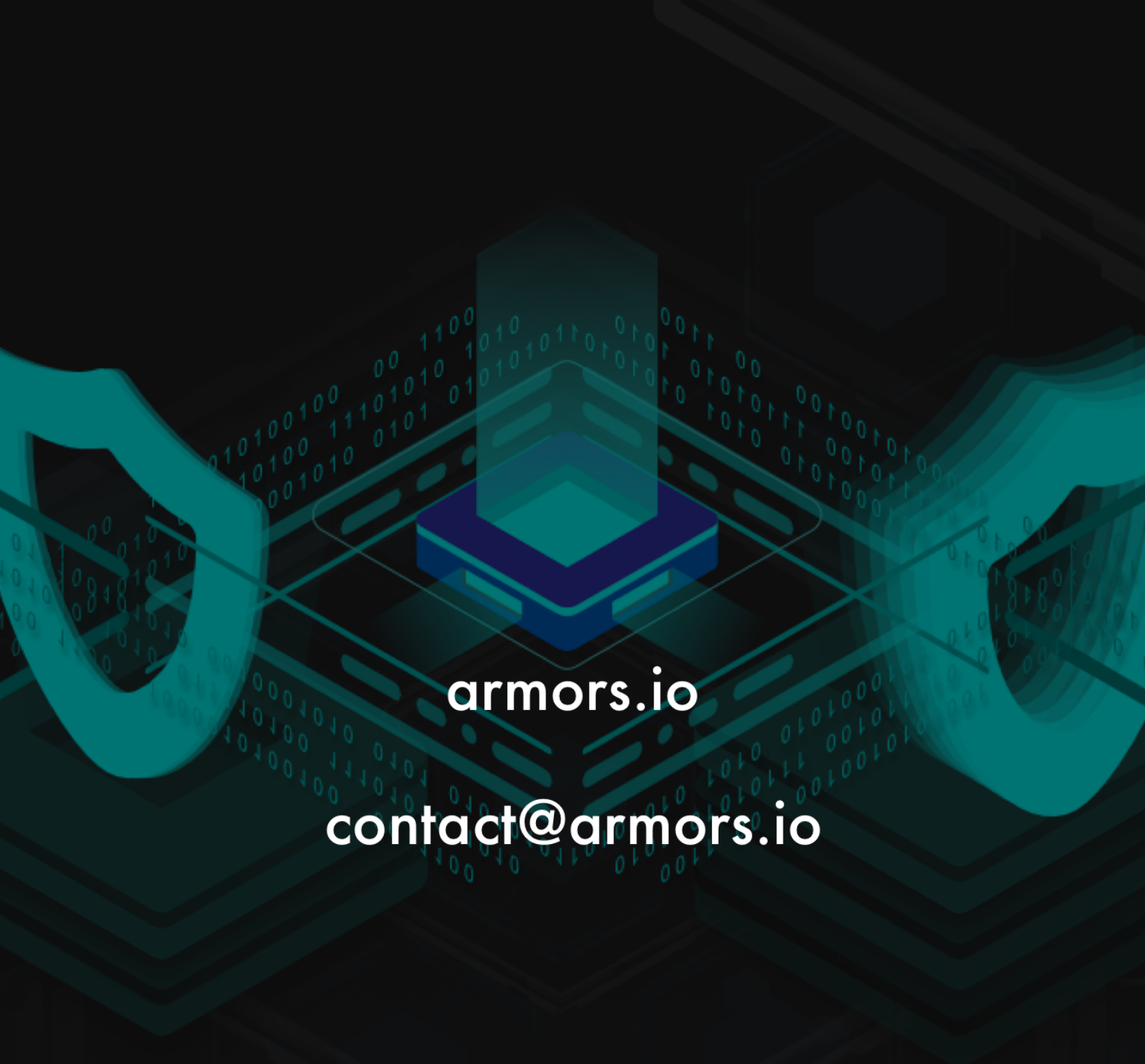
- **安全建议:** 不.

权限限制

- **说明:**
  可以控制流动性或质押池等，或对其他用户施加不合理限制的合约管理者。
- **检测结果:**

  ```
  通过!
  ```

- **安全建议:** 不.



armors.io

contact@armors.io