GAINSWAP PROJECT BACKDOOR ANALYSIS

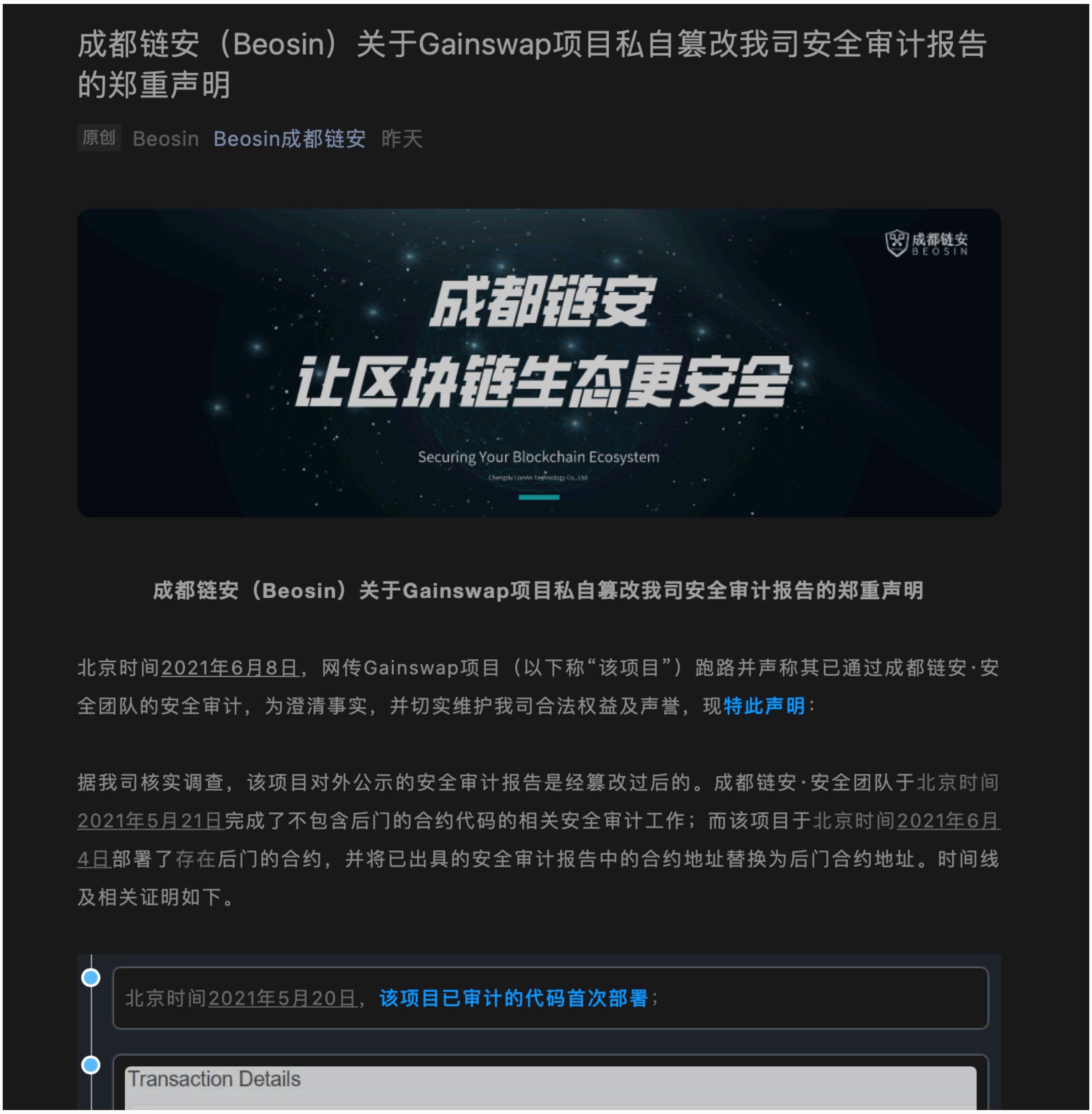
- Author: xxxeyJ
- Blog: https://tricksongs.com/

前言

最近成都链安(lianantech)在其公众媒体号发布了一则声明，大体故事的经过就是有个项目方利用链安为其审计背书，而后另起存在后门的智能合约并以此替换审计报告中的合约地址以博取信任从而为后续的跑路做准备，所以我打算简简单单吃个瓜，顺便分析一下存在问题的合约代码

这是链安方面发布的声明：

https://mp.weixin.qq.com/s/qKf4Hgg8bUEmiRoJSMXxSw


成都链安（Beosin）关于Gainswap项目私自篡改我司安全审计报告的郑重声明

原创 Beosin Beosin成都链安 昨天

成都链安
BEOSIN
成都链安
让区块链生态更安全
Securing Your Blockchain Ecosystem
Chengdu LianAn Technology Co.,Ltd.

成都链安（Beosin）关于Gainswap项目私自篡改我司安全审计报告的郑重声明

北京时间2021年6月8日，网传Gainswap项目（以下称“该项目”）跑路并声称其已通过成都链安·安全团队的安全审计，为澄清事实，并切实维护我司合法权益及声誉，现**特此声明**：

据我司核实调查，该项目对外公示的安全审计报告是经篡改过后的。成都链安·安全团队于北京时间2021年5月21日完成了不包含后门的合约代码的相关安全审计工作；而该项目于北京时间2021年6月4日部署了存在后门的合约，并将已出具的安全审计报告中的合约地址替换为后门合约地址。时间线及相关证明如下。

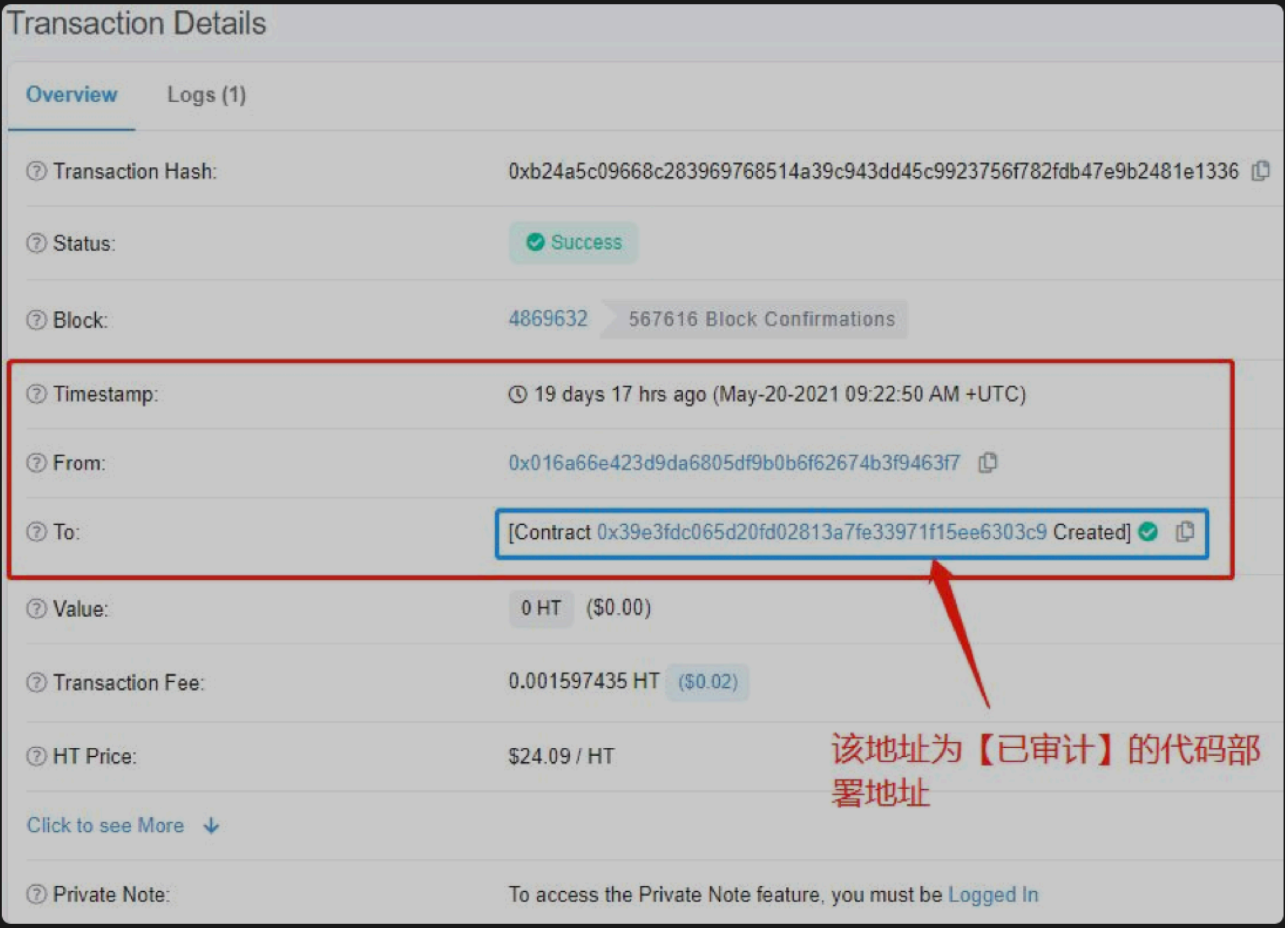
北京时间2021年5月20日，**该项目已审计的代码首次部署**；

Transaction Details

链安方面称这是通过审计无后门的智能合约:
0x39e3fdc065d20fd02813a7fe33971f15ee6303c9

合约代码: https://hecoinfo.com/address/0x39e3fdc065d20fd02813a7fe33971f15ee6303c9#code

北京时间2021年5月20日，**该项目已审计的代码首次部署**；



Transaction Details

Overview Logs (1)

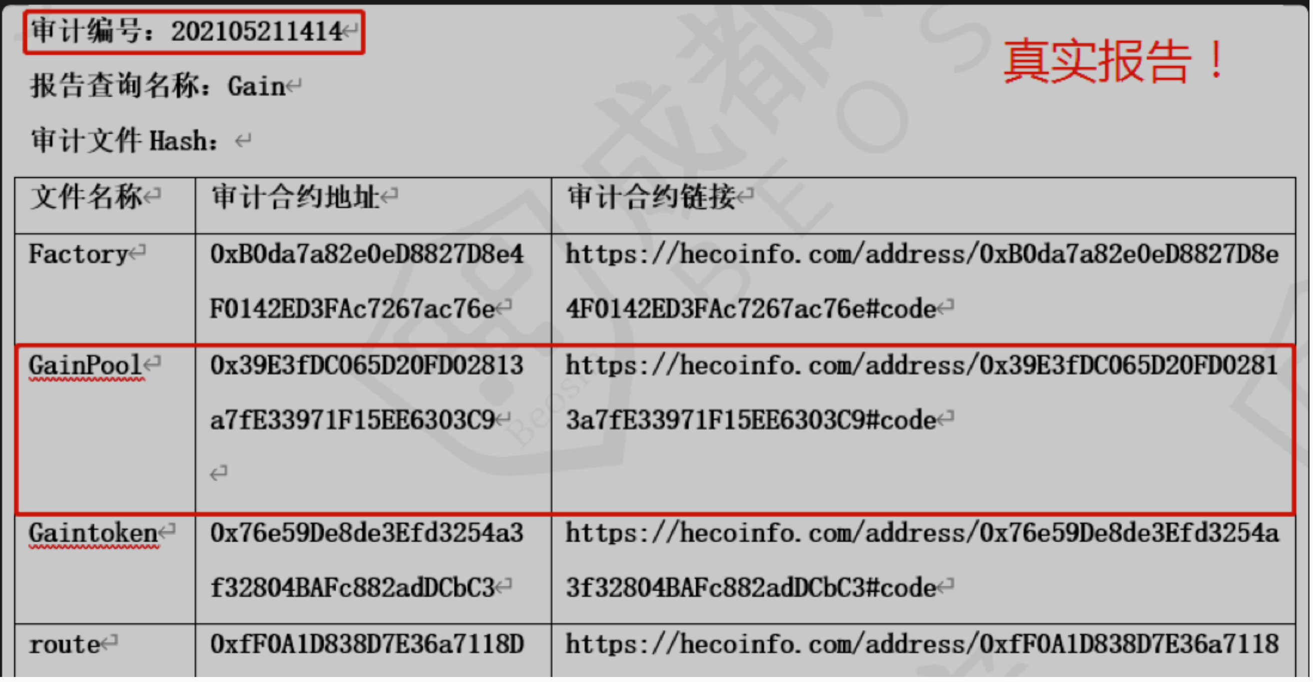
Transaction Hash:	0xb24a5c09668c283969768514a39c943dd45c9923756f782fdb47e9b2481e1336
Status:	Success
Block:	4869632 567616 Block Confirmations
Timestamp:	19 days 17 hrs ago (May-20-2021 09:22:50 AM +UTC)
From:	0x016a66e423d9da6805df9b0b6f62674b3f9463f7
To:	[Contract 0x39e3fdc065d20fd02813a7fe33971f15ee6303c9 Created]
Value:	0 HT ($0.00)
Transaction Fee:	0.001597435 HT ($0.02)
HT Price:	$24.09 / HT

Click to see More

Private Note: To access the Private Note feature, you must be Logged In

该地址为【已审计】的代码部署地址

北京时间2021年5月21日，**我司为该项目出具安全审计报告**；



审计编号：202105211414

真实报告！

报告查询名称：Gain

审计文件 Hash：

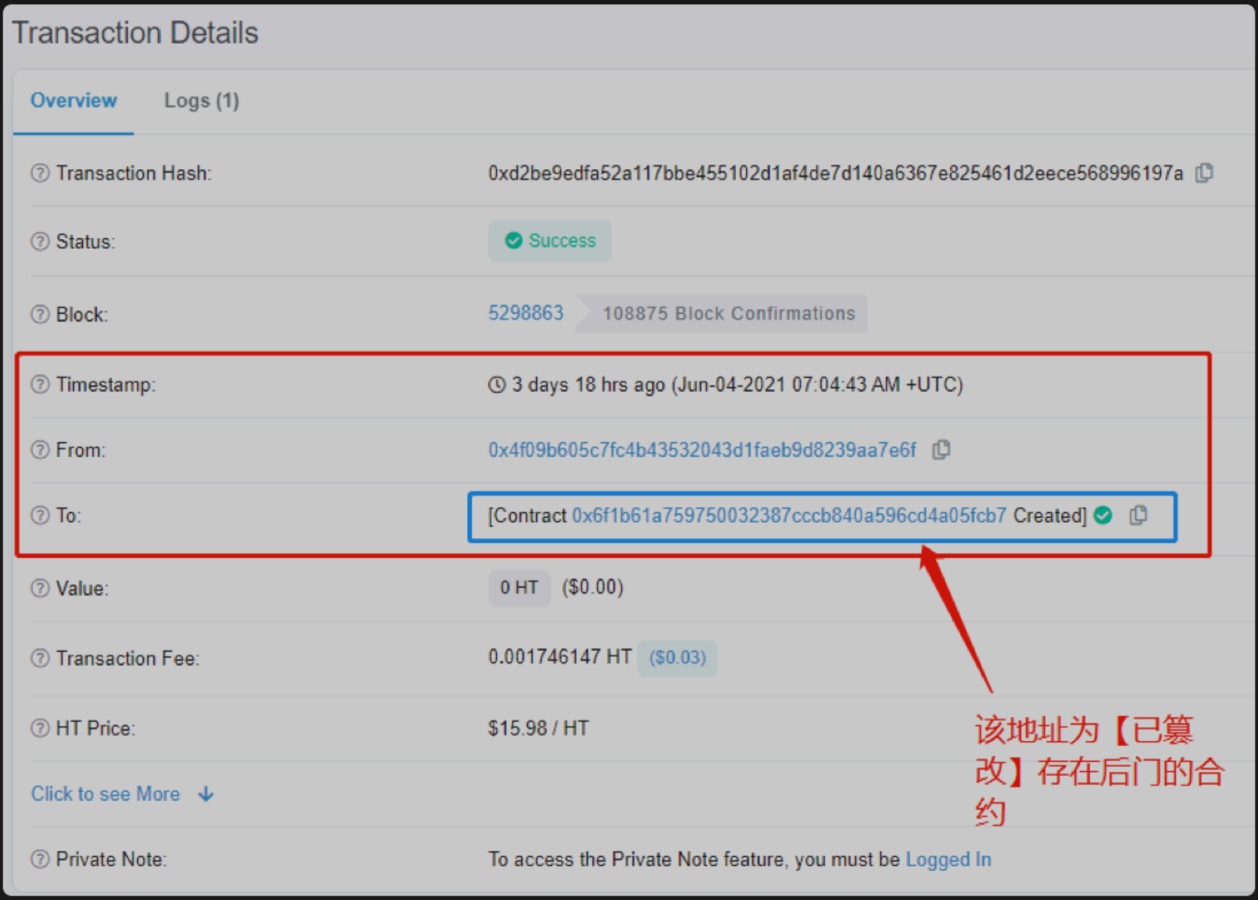
文件名称	审计合约地址	审计合约链接
Factory	0xB0da7a82e0eD8827D8e4F0142ED3FAc7267ac76e	https://hecoinfo.com/address/0xB0da7a82e0eD8827D8e4F0142ED3FAc7267ac76e#code
GainPool	0x39E3fDC065D20FD02813a7fE33971F15EE6303C9	https://hecoinfo.com/address/0x39E3fDC065D20FD02813a7fE33971F15EE6303C9#code
Gaintoken	0x76e59De8de3Efd3254a3f32804BAFc882adDCbC3	https://hecoinfo.com/address/0x76e59De8de3Efd3254a3f32804BAFc882adDCbC3#code
route	0xfF0A1D838D7E36a7118D	https://hecoinfo.com/address/0xfF0A1D838D7E36a7118

下面则是链安方面称存在后门的智能合约:

0x6f1b61a759750032387cccb840a596cd4a05fcb7

合约代码: https://hecoinfo.com/address/0x6f1b61a759750032387cccb840a596cd4a05fcb7#code

北京时间2021年6月4日，**链上记录显示，该项目私自部署存在后门的合约，并篡改由我司出具的安全审计报告中的合约地址**；

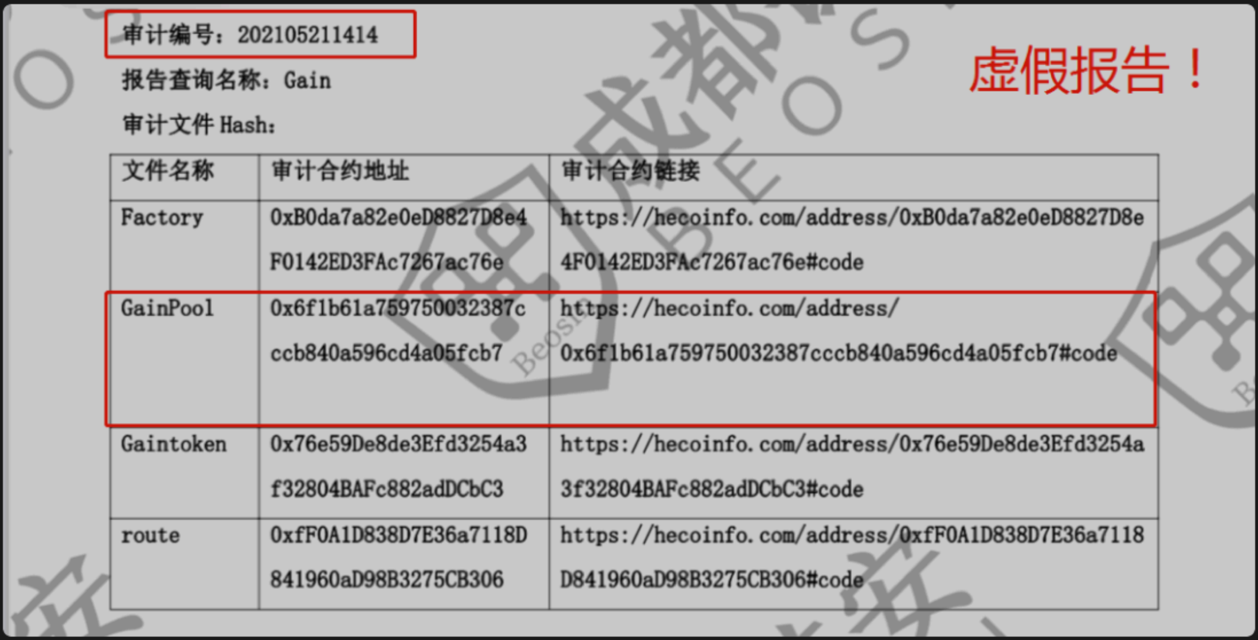


Transaction Details

Overview　Logs (1)

Transaction Hash:	0xd2be9edfa52a117bbe455102d1af4de7d140a6367e825461d2eece568996197a
Status:	Success
Block:	5298863　108875 Block Confirmations
Timestamp:	3 days 18 hrs ago (Jun-04-2021 07:04:43 AM +UTC)
From:	0x4f09b605c7fc4b43532043d1faeb9d8239aa7e6f
To:	[Contract 0x6f1b61a759750032387cccb840a596cd4a05fcb7 Created]
Value:	0 HT　($0.00)
Transaction Fee:	0.001746147 HT　($0.03)
HT Price:	$15.98 / HT
Click to see More	
Private Note:	To access the Private Note feature, you must be Logged In

该地址为【已篡改】存在后门的合约



审计编号：202105211414

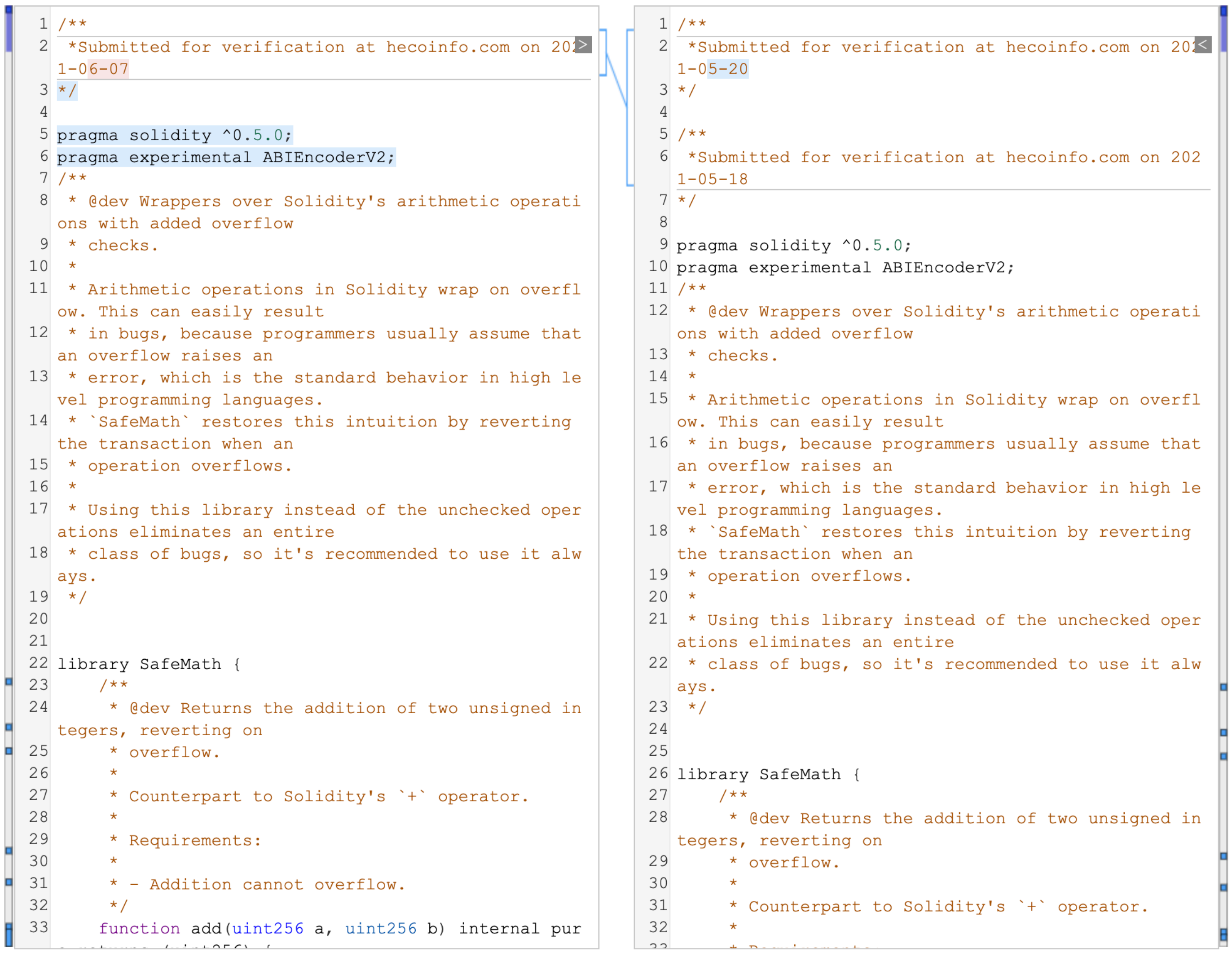
报告查询名称：Gain

审计文件 Hash：

文件名称	审计合约地址	审计合约链接
Factory	0xB0da7a82e0eD8827D8e4F0142ED3FAc7267ac76e	https://hecoinfo. com/address/0xB0da7a82e0eD8827D8e4F0142ED3FAc7267ac76e#code
GainPool	0x6f1b61a759750032387cccb840a596cd4a05fcb7	https://hecoinfo. com/address/0x6f1b61a759750032387cccb840a596cd4a05fcb7#code
Gaintoken	0x76e59De8de3Efd3254a3f32804BAFc882adDCbC3	https://hecoinfo. com/address/0x76e59De8de3Efd3254a3f32804BAFc882adDCbC3#code
route	0xfF0A1D838D7E36a7118D841960aD98B3275CB306	https://hecoinfo. com/address/0xfF0A1D838D7E36a7118D841960aD98B3275CB306#code

虚假报告！

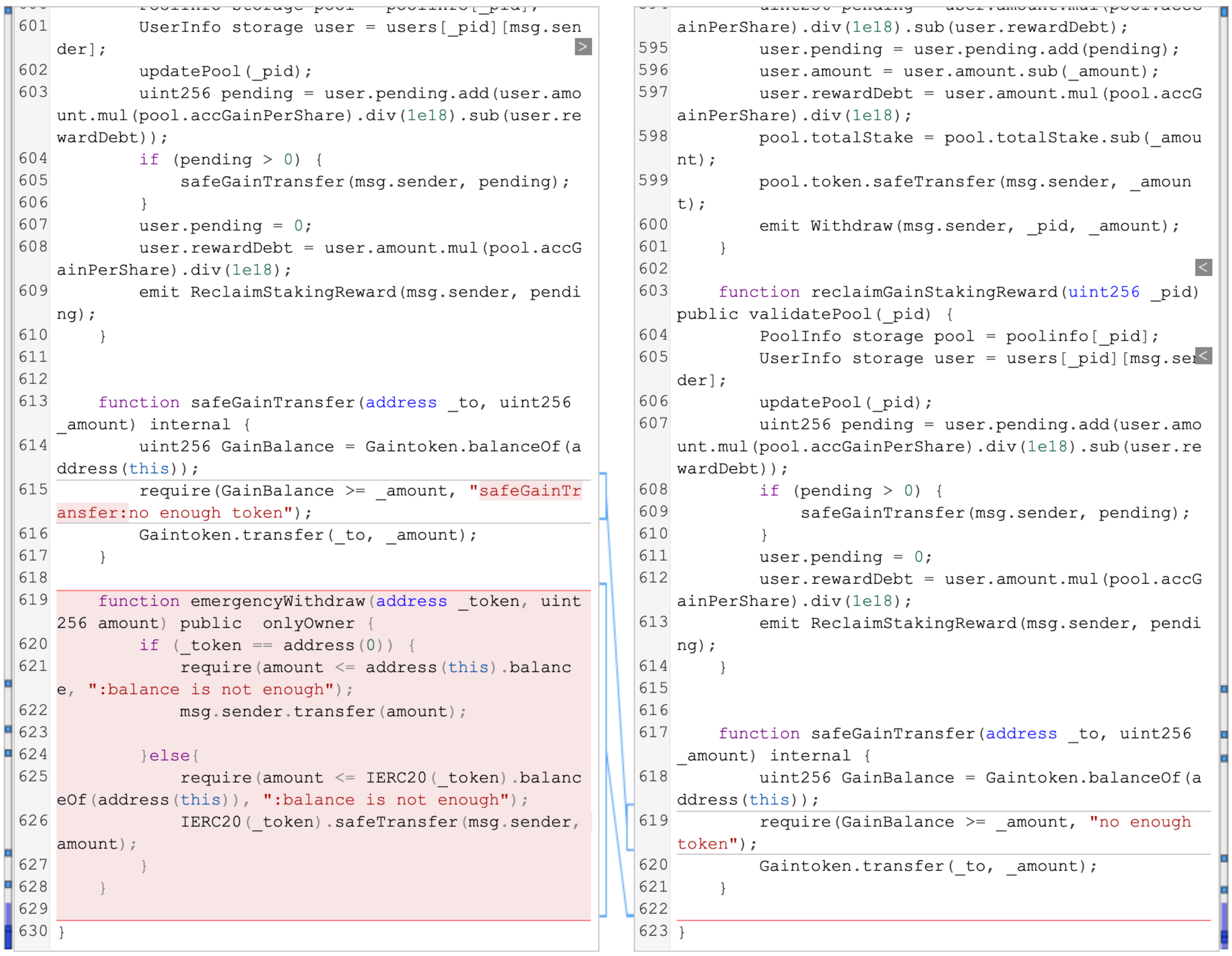
代码比对



```
/**
 *Submitted for verification at hecoinfo.com on 2021-06-07
*/

pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;
/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */


library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pur
```

```
/**
 *Submitted for verification at hecoinfo.com on 2021-05-20
*/

/**
 *Submitted for verification at hecoinfo.com on 2021-05-18
*/

pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;
/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */


library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
```

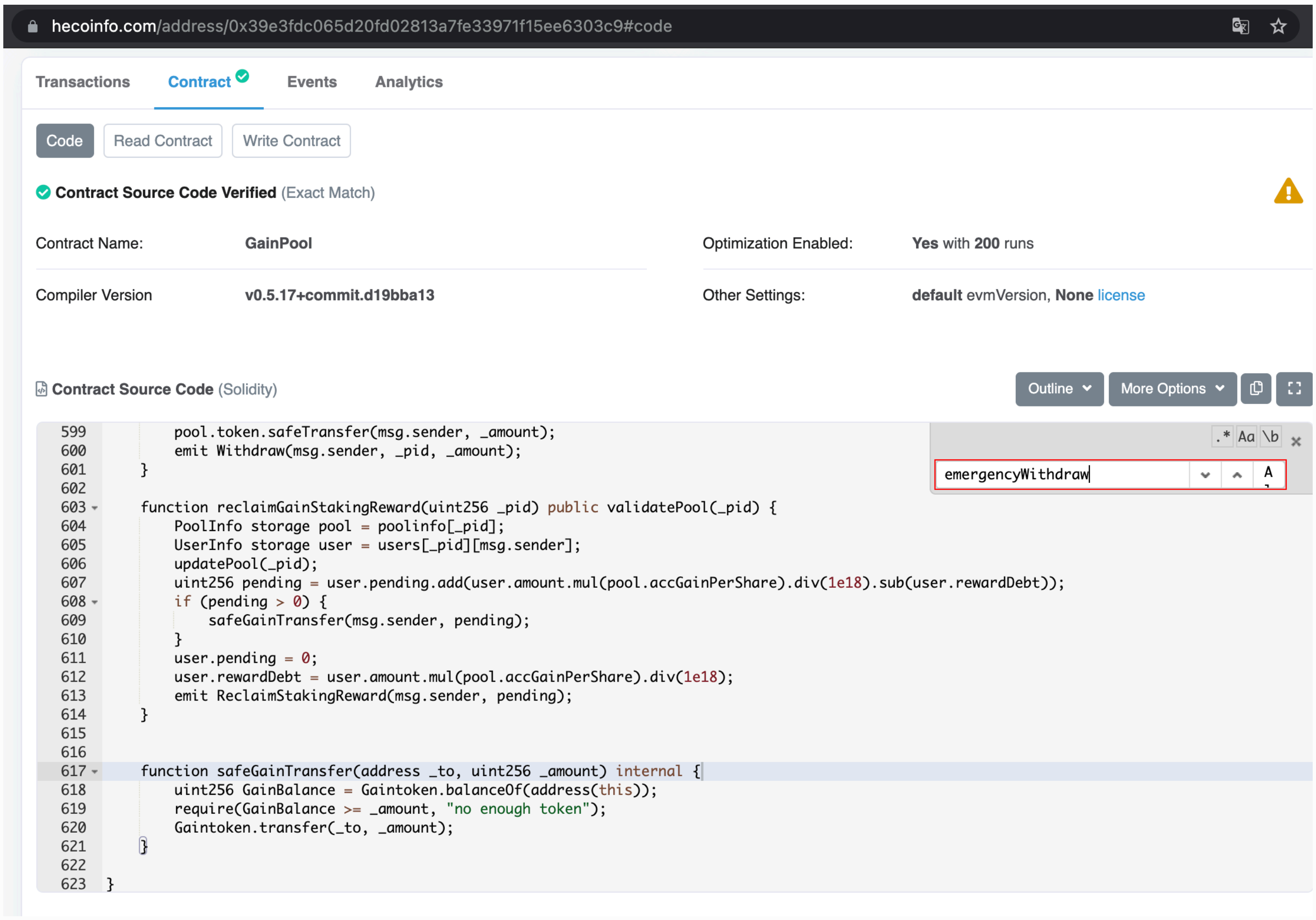
两份合约代码前后有几处小部分仅是修改了一些 `errorMessage`，存在问题的代码如下所示



```
        UserInfo storage user = users[_pid][msg.sender];
        updatePool(_pid);
        uint256 pending = user.pending.add(user.amount.mul(pool.accGainPerShare).div(1e18).sub(user.rewardDebt));
        if (pending > 0) {
            safeGainTransfer(msg.sender, pending);
        }
        user.pending = 0;
        user.rewardDebt = user.amount.mul(pool.accGainPerShare).div(1e18);
        emit ReclaimStakingReward(msg.sender, pending);
    }


    function safeGainTransfer(address _to, uint256 _amount) internal {
        uint256 GainBalance = Gaintoken.balanceOf(address(this));
        require(GainBalance >= _amount, "safeGainTransfer:no enough token");
        Gaintoken.transfer(_to, _amount);
    }

    function emergencyWithdraw(address _token, uint256 amount) public  onlyOwner {
        if (_token == address(0)) {
            require(amount <= address(this).balance, ":balance is not enough");
            msg.sender.transfer(amount);

        }else{
            require(amount <= IERC20(_token).balanceOf(address(this)), ":balance is not enough");
            IERC20(_token).safeTransfer(msg.sender, amount);
        }
    }

}
```

```
ainPerShare).div(1e18).sub(user.rewardDebt);
        user.pending = user.pending.add(pending);
        user.amount = user.amount.sub(_amount);
        user.rewardDebt = user.amount.mul(pool.accGainPerShare).div(1e18);
        pool.totalStake = pool.totalStake.sub(_amount);
        pool.token.safeTransfer(msg.sender, _amount);
        emit Withdraw(msg.sender, _pid, _amount);
    }

    function reclaimGainStakingReward(uint256 _pid) public validatePool(_pid) {
        PoolInfo storage pool = poolinfo[_pid];
        UserInfo storage user = users[_pid][msg.sender];
        updatePool(_pid);
        uint256 pending = user.pending.add(user.amount.mul(pool.accGainPerShare).div(1e18).sub(user.rewardDebt));
        if (pending > 0) {
            safeGainTransfer(msg.sender, pending);
        }
        user.pending = 0;
        user.rewardDebt = user.amount.mul(pool.accGainPerShare).div(1e18);
        emit ReclaimStakingReward(msg.sender, pending);
    }


    function safeGainTransfer(address _to, uint256 _amount) internal {
        uint256 GainBalance = Gaintoken.balanceOf(address(this));
        require(GainBalance >= _amount, "no enough token");
        Gaintoken.transfer(_to, _amount);
    }

}
```

很明显可以发现这份存在问题的合约在原有的基础上凭空多了一个 `emergencyWithdraw` function



Contract Source Code Verified (Exact Match)

Contract Name: GainPool

Optimization Enabled: Yes with 200 runs

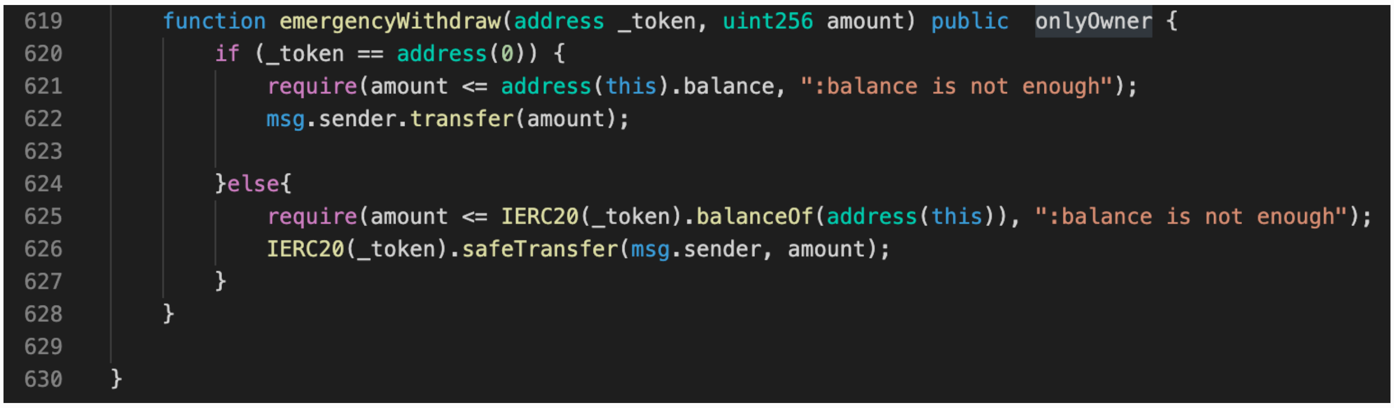
Compiler Version v0.5.17+commit.d19bba13

Other Settings: default evmVersion, None license

Contract Source Code (Solidity)

```
        pool.token.safeTransfer(msg.sender, _amount);
        emit Withdraw(msg.sender, _pid, _amount);
    }

    function reclaimGainStakingReward(uint256 _pid) public validatePool(_pid) {
        PoolInfo storage pool = poolinfo[_pid];
        UserInfo storage user = users[_pid][msg.sender];
        updatePool(_pid);
        uint256 pending = user.pending.add(user.amount.mul(pool.accGainPerShare).div(1e18).sub(user.rewardDebt));
        if (pending > 0) {
            safeGainTransfer(msg.sender, pending);
        }
        user.pending = 0;
        user.rewardDebt = user.amount.mul(pool.accGainPerShare).div(1e18);
        emit ReclaimStakingReward(msg.sender, pending);
    }


    function safeGainTransfer(address _to, uint256 _amount) internal {
        uint256 GainBalance = Gaintoken.balanceOf(address(this));
        require(GainBalance >= _amount, "no enough token");
        Gaintoken.transfer(_to, _amount);
    }

}
```

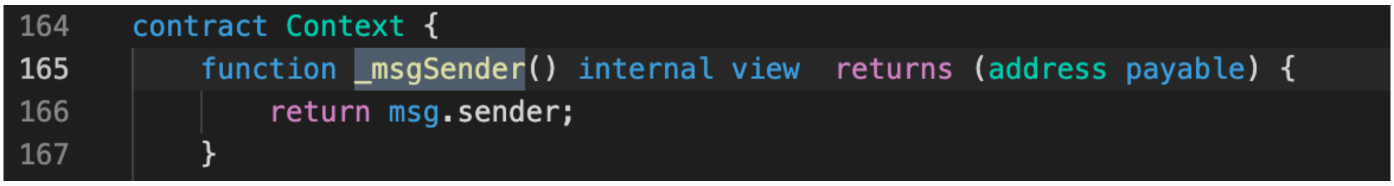
代码分析



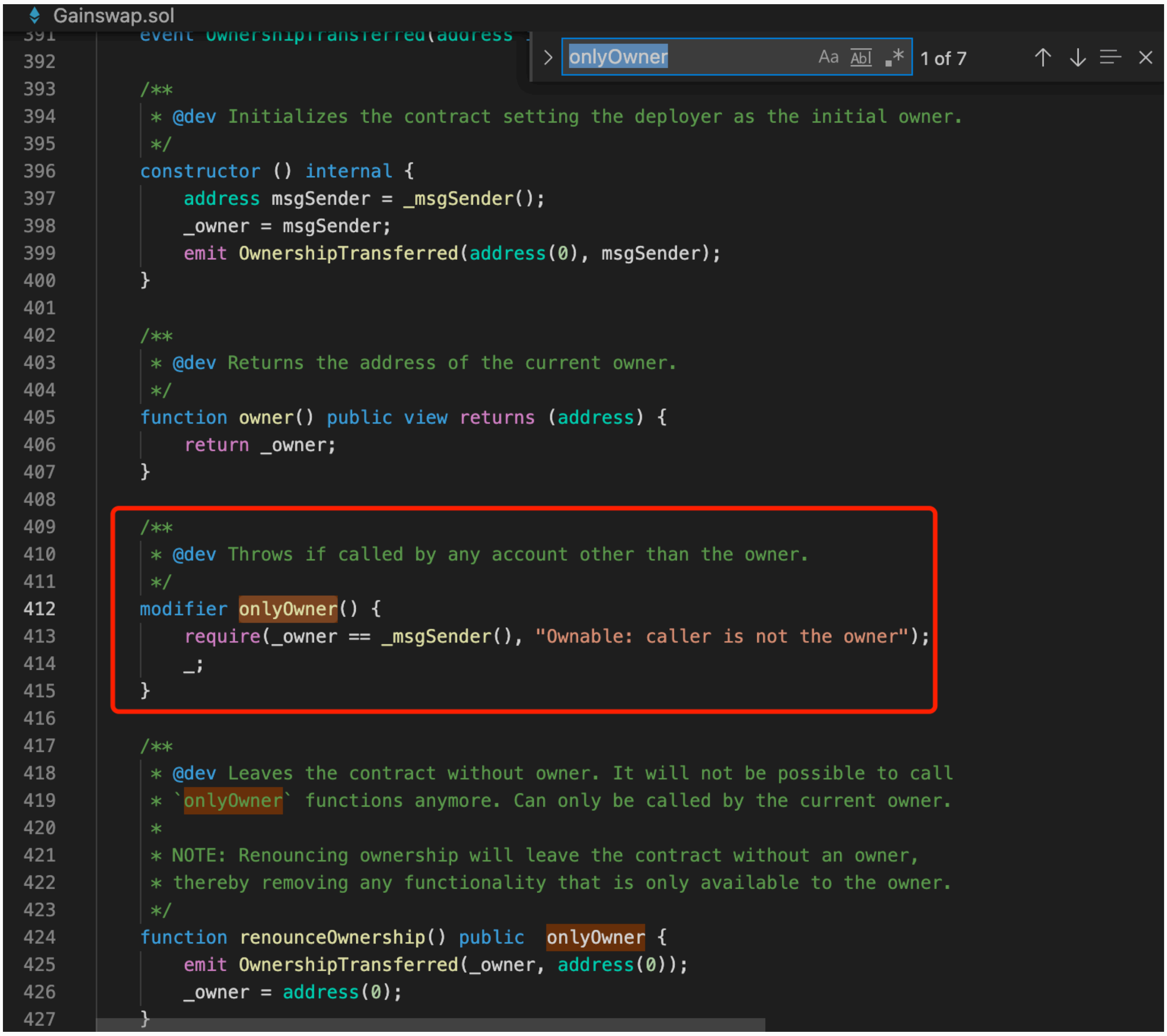
```
    function emergencyWithdraw(address _token, uint256 amount) public  onlyOwner {
        if (_token == address(0)) {
            require(amount <= address(this).balance, ":balance is not enough");
            msg.sender.transfer(amount);

        }else{
            require(amount <= IERC20(_token).balanceOf(address(this)), ":balance is not enough");
            IERC20(_token).safeTransfer(msg.sender, amount);
        }
    }

}
```

`emergencyWithdraw` 这一函数的作用分析如下，函数体使用了 `onlyOwner` 这个修饰器对函数进行修饰

`onlyOwner` 修饰器的定义如下，使用 `require` 方法进行条件判断， `_msgSender` 函数的返回值为 `msg.sender` ，校验当前调用者是否为合约的 `Owner` 账户



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



Gainswap.sol

onlyOwner Aa Abl .* 1 of 7

```
    event OwnershipTransferred(address 

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

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

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

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

而后通过修饰器校验后进而执行函数体代码，调用此函数需传递两个参数， `address _token` 参数位直接传 `0x0000000000000000000000000000000000000000` 即可进入 `if-true` 的代码段， `uint amout` 用于传递当前合约中的余额，else 代码段用于提取 Token，只要满足当前合约 address(this) 在目标智能合约Contract::balanceOf 这个金额小于等于的条件，就可以直接把当前合约中指定的(`if-true`)原生币 `HT` 或者 Token 金额转给 `msg.sender` 消息发起者了，从而实现了项目方提款跑路这一操作

:)