June 30, 2021

GUARDEN

HIFarm Security Assessment

ABOUT GUARDEN

Guarden is an industry-leading blockchain security company composed of top talents in the computer field from world-renowned universities. It mainly provides security audit and accuracy certification services based on smart contracts and blockchain protocols. We have the world's top technology professionals in blockchain research and development and our unique and most stringent audit technology, which can provide users with the most reliable and trustworthy security audit services in the world.

Guarden is a company focused on blockchain security, and it also has a lot of experience in traditional network security and attack-defence experiences. The team members are all security technology members with more than 15 years. Team members are also blockchain technology experts and they have very rich experience in blockchain security. Guarden.org will be the most indestructible shield of your project.

Through the most stringent audit specifications and a variety of audit tools, combined with the best audit engineers, we will provide the most professional and effective audit services and security for your project.

Website : Guarden.Org

Table of Contents

Summary

In this report, we conduct a security audit on the smart contract code of the HIFarm project. Our purpose is to discover security issues and vulnerabilities in the smart contract code in the project, as well as the problem of third-party dependent libraries that are potentially problematic.

Our company uses static analysis, dynamic analysis, typical case testing, code review and manual review to conduct comprehensive and multi-angle security testing on the three aspects of HIFarm project smart contract code standardization, security and business logic.

HIFarm is a revenue aggregation platform based on Binance Smart Chain. We have received HIFarm's audit application. After communicating with HIFarm, we will focus on the following key points during the audit:

A. View all types of attacks that have occurred so far to test smart contracts

B. Construct potential attack types to test smart contracts

C. According to the current industry's most professional code specifications, to review smart contracts

D. Invite industry's top experts to conduct a comprehensive review of smart contracts

Overview

Project Summary

We audited the smart contract code of the HIFarm.Finance project. The specific project information is as follows:

HIfarm is a safe and efficient yield aggregation platform.

The HIlarm protocol empowers farmers to leverage their yield-seeking tendencies to optimize yield compounding strategy on Binance Smart Chain.

Project contract address: https://github.com/HiFarm/Contract

Audit commit: 0922d817cec29b94a22bb94c2858530f1c64a925

Project Structure

```
./comptroller
├── Comptroller.sol
├── ComptrollerInterface.sol

./dashboard
├── DashBoard.sol

./external
├── chainlink
|   └── AggregatorV3Interface.sol
├── pancake
|   └── IMasterChef.sol
```

```
|   └── IPancakeFactory.sol
|   └── IPancakePair.sol
|   └── IPancakeRouter01.sol
|   └── IPancakeRouter02.sol

./governance
├── Governor.sol
├── Timelock.sol

./libraries
├── FixedPoint.sol
├── Helper.sol
├── HomoraMath.sol
├── OwnerPausableUpgradeable.sol
├── TransferHelper.sol

./pool
├── HIFarmRewardTokenPool.sol
├── Pool.sol
├── PoolInterface.sol
├── RewardTokenFarmPool.sol
├── ShareTokenFarmPool.sol
├── TokenPool.sol
├── defiswap
|   └── CakeFarmPancakePool.sol
|   └── LPToCakeFarmPancakePool.sol
```

```
./price
├── PriceInterface.sol
├── PriceProvider.sol

./token
├── HIFarmToken.sol

./zap
├── Zap.sol
├── ZapInterface.sol
```

Contract Structure

HIFarm.Finance is mainly composed of nine parts, namely: comptroller, dashboard, external, governance, libraries, pool, price, token, zap. The main contract logic is in Pool.sol, TokenPool.sol, ShareTokenFarmPool.sol, RewardTokenFarmPool. sol, HIFarmRewardTokenPool.sol, CakeFarmPancakePool.sol, LPToCakeFarmPancakePool.sol, Comptroller.sol, Zap.sol, PriceProvider.sol, HIFarmToken.sol.

The user mortgages the token to the corresponding pool by calling the depositTo interface in the corresponding pool to mortgage the corresponding token to the corresponding contract. The contract will automatically mortgage the token to the corresponding swap to obtain income. At the same time, it will perform an automatic reinvestment function to help users improve APY.

Audit Method

We will adopt systematic, professional and rigorous audit methods to conduct all aspects of the audit of smart contracts. The following methods are the solutions we adopted in this audit. The specific methods are as follows:

METHOD	CHECK
Code Review	Compiler version specification
	Redundant code specification
	SafeMath functional specification
	Require/assert usage specification
	Gas Limit and Loops
	Falback function usage specification
	Visibility Specification
	Deprecated Item Specification
	Data Consistency
	Repository Consistency
	Deployment Consistency
	Unsafe type inference
	Unchecked math
	Unchecked external call
	ERC20 API violation
	Costly Loop
	Style guide violation
	Transaction-Ordering Dependence
	DoS with Block Gas Limit
	DoS with (Unexpected) Throw
	Timestamp Dependence
	Ownership Takeover

METHOD	CHECK
	Reentrancy
	Performance optimization audit
Vulnerability audit	Integer overflow audit
	Reentrancy attack audit
	Pseudo-random number generation audit
	Transaction order depends on audit
	Denial of service attack audit
	Function call permission audit
	call/delegatecall security audit
	Return value security audit
	tx.origin uses security audit
	Replay attack audit
	Variable coverage audit
	Overflow audit
	Access control audit
	Excessive authority audit
	Hard-coded address security
	Show coding security
	Exception verification audit
	Rollback attack audit
	False notice audit
	False error notification audit
	Counterfeit currency audit
	Dust attack security audit
	Microfork security audit
	Crowd-out attack security audit
Business audit	Business logic audit

METHOD	CHECK
	Business develop audit
	Black box testing
	Grey box testing
	White box testing

Audit Introduction

In response to different security issues on the blockchain, we have classified the issues as follows, and the details are as follows:

CRITICAL	Critical vulnerabilities will have a significant impact on the security of the blockchain. It is strongly recommended to fix the critical vulnerabilities, otherwise it will seriously affect the development of the project.
MAJOR	Major vulnerabilities will affect the normal operation of the blockchain. It is recommended to repair the vulnerabilities of medium risk.
MEDIUM	Medium vulnerabilities may affect the operation of smart contracts in some special scenarios. It is recommended that the project party evaluate whether these vulnerabilities need to be repaired according to the actual situation.
MINOR	From the theoretical analysis, there is the possibility of safety hazards, but it is extremely difficult to reproduce such hazards in engineering.
TIPS	There can be better coding or architecture practices, and the project party can consider upgrading.

Audit Scope

FILE	MD5
Comptroller.sol	915e97e4b4029c13bb77761034c8ef33
DashBoard.sol	626c330842f9175973c31fa9f59c2ea2
Governor.sol	5c2da75ce5e8308562db44d3fc6e225c
Timelock.sol	e72b0ee2b306c0f2270ef0c2aa19ecf5
FixedPoint.sol	7238e43c3e340958c58b44432b91d93b
Helper.sol	5018c5a18a32088d493846df0ab97f30
HomoraMath.sol	2f9e93168a6747057775855b4e7e98d0
TransferHelper.sol	b312dfef05a1e7fdbade150348591bd0
HIFarmRewardTokenPool.sol	acee0a7e48f8902e2c5ba476c1d3a16a
Pool.sol	9159b50f83180428b07a944fb64e5f3c
RewardTokenFarmPool.sol	3deb4e2e37a55209d80d59a9bdbbdca4
ShareTokenFarmPool.sol	88a38400589a48a5c689cc809af46ff6
TokenPool.sol	2564e5c31e2695218ec00e637d904e81
CakeFarmPancakePool.sol	8c4cd6aef76e8b27398b79a8bb08cdbd
LPToCakeFarmPancakePool.sol	9e4a5f6988a71d884b023ab95a98f0b3
PriceProvider.sol	541dd790a0c07f523a786fc4bfce4284
HIFarmToken.sol	ace6057755275ec313bc8a2982def4bd
Zap.sol	e115e5b50ae9fe5a304c0308f1aeec78

Code Analysis

Comptroller Contract

Analysis of Rights Management

This contract carries out the management contract of the HIF Token. The highest authority administrator of the contract is the Owner, and the default is the deployer of the contract.

Related functions controlled by Owner: setDev, setHIFToken, setHIFPool, setPriceProvider, setWhiteList, addMarket, setMarketAccPerBlock, setMarketPaused, setExtendRewardsDuration, setExchangeRateMantissa, mintNewFarmReward

Audit recommendation: None.

Audit result: authority control verification passed.

HIF Token Additional Issuance Management

This contract simultaneously carries out the additional issuance of HIF Token and the HIF Token exchange management of the performance fee. The management of the additional issuance of HIF Token uses the markets variable for control and management. The corresponding data structure is as follows:

```
struct Market {
    uint256 accPerSecond;
    uint256 accPerShareMantissa;
    uint256 lastTime;
    bool isListed;
    bool isPaused;
}
```

The rewards for the additional issuance of HIF Token are mainly divided into two parts, the first part is the overall update;

```
    function _updateReward(address _pool) internal {
        Market storage market = markets[_pool];
        market.accPerShareMantissa = rewardPerShare(_pool);
        market.lastTime = block.timestamp;
        marketExtendRewardInfos[_pool].lastTime = lastTimeExtendRewardApplicable(_pool);
    }
    function rewardPerShare(address _pool) public view returns (uint256) {
        Market storage market = markets[_pool];
        uint256 blockTime = block.timestamp;
        uint256 deltaTimes = blockTime.sub(market.lastTime);
        uint256 accPerShareMantissa = market.accPerShareMantissa;
        uint256 accrued = 0;
        if (deltaTimes > 0 && market.accPerSecond > 0) {
            accrued = deltaTimes.mul(market.accPerSecond);
        }
        ExtendRewardInfo storage extendRewardInfo = marketExtendRewardInfos[_pool];
        // check?
        uint256 extendDeltaTimes =
lastTimeExtendRewardApplicable(_pool).sub(extendRewardInfo.lastTime);
        if (extendDeltaTimes > 0) {
            accrued = accrued.add(extendDeltaTimes.mul(extendRewardInfo.accPerSecond));
        }
        if (accrued > 0) {
            uint256 measureTotalSupply = PoolInterface(_pool).totalShare();
            if (measureTotalSupply > 0) {
                uint256 newAccPerShareMantissa = FixedPoint.calculateMantissa(accrued,
measureTotalSupply);
                accPerShareMantissa = accPerShareMantissa.add(newAccPerShareMantissa);
```

```
            }
        }

        return accPerShareMantissa;
    }
```

The second part is the user's reward update.

```
    function _distributeNewForUser(address _pool, address _user) internal {
        Market storage market = markets[_pool];
        UserState storage userState = userStates[_pool][_user];
        if (market.accPerShareMantissa == userState.lastAccPerShareMantissa) {
            return;
        }
        uint256 deltaAccPerShareMantissa =
market.accPerShareMantissa.sub(userState.lastAccPerShareMantissa);
        uint256 userMeasureBalance = PoolInterface(_pool).shareOf(_user);
        uint256 newAccrued = FixedPoint.multiplyUintByMantissa(userMeasureBalance,
deltaAccPerShareMantissa);
        userState.accrued = userState.accrued.add(newAccrued);
        userState.lastAccPerShareMantissa = market.accPerShareMantissa;
    }
```

Audit recommendation: None.

Audit result: The logic of reward distribution is in line with expectations, and the review passed.

HIFarmRewardTokenPool Contract

Analysis of Recharge, Claim reward, Cash Withdrawal

This contract is to mortgage HIF Token into the contract, and you can receive additional issuance HIF Token or Busd rewards. The three most important functions of this contract are the recharge, the claim reward, and the withdrawal (redeem) interface. The main logic is as follows:

```
    function _supply(address minter, uint256 amount) internal override returns (uint256) {
        comptroller.beforeSupply(minter, amount);
        _updateReward();
        _distributeNewForUser(minter);

        UserStakeInfo storage userStake = accountStake[minter];
        totalSupply = totalSupply.add(amount);
        userStake.amount = userStake.amount.add(amount);
        userStake.lastTimestamp = _currentTime();

        uint256 stakeAmount = _stakeToFarm(amount);
        return stakeAmount;
    }

    function _claimReward(address user) internal virtual {
        _updateReward();
        _distributeNewForUser(user);
        UserRewardState storage userRewardState = userRewardStates[user];
        uint256 accruedAmount = userRewardState.accrued;
        userRewardState.accrued = 0;
        uint256 amount = _convertAccrued(accruedAmount);

        uint256 performanceFeeAmount = amount > 0 ?
FixedPoint.multiplyUintByMantissa(amount, performanceFeeFactorMantissa) : 0;
        performanceFeeAmount = _dealWithPerformanceFee(user, performanceFeeAmount);
        amount = amount.sub(performanceFeeAmount);
```

```
        uint256 rewardAmount = _transferOutReward(user, amount);
        emit ClaimReward(user, rewardAmount);
        //claim
        comptroller.claim(user);
    }

    function _redeem(address user, uint256 amount) internal override returns (uint256) {
        require(amount <= accountStake[user].amount, "insufficient user balance");
        require(amount <= totalSupply, "insufficient total supply");

        comptroller.beforeRedeem(user, amount);
        _updateReward();
        _distributeNewForUser(user);
        totalSupply = totalSupply.sub(amount);
        accountStake[user].amount = accountStake[user].amount.sub(amount);

        uint256 outAmount = _unStakeForWithdraw(amount);
        uint256 withdrawFeeAmount = outAmount > 0 ? estimateWithdrawalFee(user, outAmount)
: 0;
        withdrawFeeAmount = _dealWithWithdrawFee(withdrawFeeAmount);
        outAmount = outAmount.sub(withdrawFeeAmount);
        return outAmount;
    }

    function _notifyReward(uint256 amount) internal {
        _updateReward();

        if (block.timestamp >= rewardInfo.lastPeriodEndTime) {
            rewardInfo.accPerSecond = amount.div(rewardInfo.duration);
```

```
        } else {
            uint256 remaining = rewardInfo.lastPeriodEndTime.sub(block.timestamp);
            uint256 leftover = remaining.mul(rewardInfo.accPerSecond);
            rewardInfo.accPerSecond = amount.add(leftover).div(rewardInfo.duration);
        }

        uint256 rewardBalance = _rewardBalance();
        require(rewardInfo.accPerSecond <= rewardBalance.div(rewardInfo.duration),
"RewardTokenFarmPool: accPerSecond error");
        rewardInfo.lastTime = block.timestamp;
        rewardInfo.lastPeriodEndTime = block.timestamp.add(rewardInfo.duration);
        emit ReceiveReward(amount);
    }
```

Audit recommendation: None.

Audit result: The interface business logic meets expectations, and the audit passed.

CakeFarmPancakePool Contract

Analysis of Recharge, Claim reward, Cash Withdrawal

This contract is to mortgage Cake to the contract, and you can receive the rewards of Cake and HIF Token. The three most important functions in the contract are recharge, claim the reward and cash withdrawal. The specific implementation is as follows:

```
    function _supply(address minter, uint256 amount) internal virtual override returns (uint256) {
        comptroller.beforeSupply(minter, amount);
        uint256 stakeAmount = _supplyAllowed(amount);
        uint256 mintShare = shareOfAmount(stakeAmount);
        UserStakeInfo storage userStake = accountStake[minter];
```

```
        userStake.share = userStake.share.add(mintShare);
        totalShare = totalShare.add(mintShare);

        //if minter is in whitelist (pools), only interact with withdrawShare
        if (!comptroller.isInWhiteList(minter)) {
            userStake.principal = userStake.principal.add(stakeAmount);
            userStake.lastTimestamp = _currentTime();
        }

        stakeAmount = _stakeToFarm(stakeAmount);
        return stakeAmount;
    }

    function _redeemSeparate(address user, uint256 principalAmount, uint256 profitAmount)
internal virtual returns (uint256) {
        uint256 userBalance = _balanceOf(user);
        UserStakeInfo storage userStake = accountStake[user];
        principalAmount = MathUpgradeable.min(principalAmount, userStake.principal);
        uint256 amount = principalAmount.add(profitAmount);
        amount = MathUpgradeable.min(amount, userBalance);
        comptroller.beforeRedeem(user, amount);
        share = MathUpgradeable.min(share, userStake.share);
        userStake.share = userStake.share.sub(share);
        totalShare = totalShare.sub(share);

        uint256 share = shareOfAmount(amount);
        userStake.principal = userStake.principal.sub(principalAmount);
        //cleanup dust
        if (userStake.principal == 0 && userStake.share > 0 && userStake.share < DUST) {
            totalShare = totalShare.sub(userStake.share);
```

```
            userStake.share = 0;
        }

        uint256 outAmount = _unStakeForWithdraw(amount);
        uint256 withdrawFeeAmount = principalAmount > 0 ? estimateWithdrawalFee(user,
principalAmount) : 0;
        uint256 performanceFeeAmount = profitAmount > 0 ?
FixedPoint.multiplyUintByMantissa(profitAmount, performanceFeeFactorMantissa) : 0;
        uint256 dealWithFeeAmount = _dealWithFee(user, withdrawFeeAmount,
performanceFeeAmount);
        outAmount = outAmount.sub(dealWithFeeAmount);
        return outAmount;
    }

    function claimReward() external virtual override notContract whenNotPaused nonReentrant {
        address user = msg.sender;
        uint256 profit = earned(user);
        uint256 amount = _redeemSeparate(user, 0, profit);

        _doTransferOut(_stakedToken(), user, amount);
        emit ClaimReward(user, amount);
        //comptroller claim
        comptroller.claim(user);
    }
```

Audit recommendation: In the _redeemSeparate function, before calling shareOfAmount, do not modify the totalShare, otherwise it will affect the accuracy of the calculation.

Audit result: There is a problem, and the project party has been notified to fix it.

Audit Findings

FUNCTION	CHECK	LIST	RESULT
Deposit HIF Token, Farm HIF Token	Deposit	deposit minimum 0.00000001 HIF	Pass
		deposit 1 HIF	Pass
		User a deposit 1 HIF, same user again deposit 1HIF	Pass
		User a deposit 1 HIF, user b deposit 1HIF	Pass
	Claim Reward	User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user b deposits 1 HIF	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user a claim reward	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 60 seconds later, user a claim reward, click claim again after 60 seconds	Pass
		User a isn't deposited, click claim reward	Pass
		After user a deposit 1HIF, withdraw the principal, then claim reward, and charge the withdrawal fee	Pass
		claim reward, and no withdrawal fee	Pass
	Withdraw	User a deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user b deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF, user a withdraw 10HIF	Pass
		User a isn't deposited, and withdraw	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
	WithdrawAll	User a isn't deposited, and withdraw & claim	Pass
		If there is only profit and no principal, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim, click withdraw & claim again	Pass

FUNCTION	CHECK	LIST	RESULT
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
Deposit HIF-BUSD FLIP, Farm HIF	Deposit	deposit minimum 0.00000001 HIF	Pass
		deposit 1 HIF	Pass
		User a deposit 1 HIF, same user again deposit 1HIF	Pass
		User a deposit 1 HIF, user b deposit 1HIF	Pass
	Claim Reward	User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user b deposits 1 HIF	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user a claim reward	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 60 seconds later, user a claim reward, click claim again after 60 seconds	Pass
		User a isn't deposited, click claim reward	Pass
		After user a deposit 1HIF, withdraw the principal, then claim reward, and charge the withdrawal fee	Pass
		claim reward, and no withdrawal fee	Pass
	Withdraw	User a deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user b deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF, user a withdraw 10HIF	Pass
		User a isn't deposited, and withdraw	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
	WithdrawAll	User a isn't deposited, and withdraw & claim	Pass
		If there is only profit and no principal, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim, click withdraw & claim again	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass

FUNCTION	CHECK	LIST	RESULT
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
Deposit HIF, Farm BUSD	Deposit	deposit minimum 0.00000001 HIF	Pass
		deposit 1 HIF	Pass
		User a deposit 1 HIF, same user again deposit 1HIF	Pass
		User a deposit 1 HIF, user b deposit 1HIF	Pass
	Claim Reward	User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user b deposits 1 HIF	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user a claim reward	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 60 seconds later, user a claim reward, click claim again after 60 seconds	Pass
		User a isn't deposited, click claim reward	Pass
		After user a deposit 1HIF, withdraw the principal, then claim reward, and charge the withdrawal fee	Pass
		claim reward, and no withdrawal fee	Pass
	Withdraw	User a deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user b deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF, user a withdraw 10HIF	Pass
		User a isn't deposited, and withdraw	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
	WithdrawAll	User a isn't deposited, and withdraw & claim	Pass
		If there is only profit and no principal, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim, click withdraw & claim again	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass

FUNCTION	CHECK	LIST	RESULT
Deposit CAKE, Farm CAKE + HIF	Deposit	deposit minimum 0.00000001 HIF	Pass
		deposit 1 HIF	Pass
		User a deposit 1 HIF, same user again deposit 1HIF	Pass
		User a deposit 1 HIF, user b deposit 1HIF	Pass
	Claim Reward	User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user b deposits 1 HIF	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user a claim reward	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 60 seconds later, user a claim reward, click claim again after 60 seconds	Pass
		User a isn't deposited, click claim reward	Pass
		After user a deposit 1HIF, withdraw the principal, then claim reward, and charge the withdrawal fee	Pass
		claim reward, and no withdrawal fee	Pass
	Withdraw	User a deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user b deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF, user a withdraw 10HIF	Pass
		User a isn't deposited, and withdraw	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
	WithdrawAll	User a isn't deposited, and withdraw & claim	Pass
		If there is only profit and no principal, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim, click withdraw & claim again	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass

FUNCTION	CHECK	LIST	RESULT
Deposit CAKE-BNB FLIP, Farm CAKE + HIF	Deposit	deposit minimum 0.00000001 HIF	Pass
		deposit 1 HIF	Pass
		User a deposit 1 HIF, same user again deposit 1HIF	Pass
		User a deposit 1 HIF, user b deposit 1HIF	Pass
	Claim Reward	User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user b deposits 1 HIF	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 6 seconds later, user a claim reward	Pass
		User a deposit 1 HIF, set to reward1 HIF per second, 60 seconds later, user a claim reward, click claim again after 60 seconds	Pass
		User a isn't deposited, click claim reward	Pass
		After user a deposit 1HIF, withdraw the principal, then claim reward, and charge the withdrawal fee	Pass
		claim reward, and no withdrawal fee	Pass
	Withdraw	User a deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user b deposit 10 HIF, user a withdraw 1HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF	Pass
		User a deposit 10 HIF, user a withdraw 10HIF, user a withdraw 10HIF	Pass
		User a isn't deposited, and withdraw	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass
	WithdrawAll	User a isn't deposited, and withdraw & claim	Pass
		If there is only profit and no principal, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim	Pass
		User a deposit 1 HIF, after 60 seconds, accumulate profit, click withdraw & claim, click withdraw & claim again	Pass
		A withdrawal fee of 0.5% of the principal will be charged for withdrawal within 72 hours	Pass
		No withdrawal fee will be charged for withdrawals over 72 hours	Pass

Conclusion

Audit Result: PASS

Audit Date: JUN 30, 2021

Audit Team: Guarden Team

Conclusion: After communicating and cooperating with the HIFarm team, all the problems found have been fixed, and all the above risks have been eliminated by the HIFarm team. Comprehensive assessment, HIFarm no longer has the above risks and meets the audit requirements.

Disclaimer

This report is not intended as investment advice, nor as a guarantee of no code errors and any business model. It cannot be used in any way to make investment or participate in any project decision-making basis, and cannot be used as any type of investment advice. This report is to help the project party improve the quality of the code and reduce the possible risks of smart contracts. This report is only for audits conducted within the scope of the audit type given in the report, and other unknown vulnerabilities are not within the scope of the audit of this report. This report shall not be modified, transmitted, or mentioned for any purpose without the written authorization of Guarden. Any audit company will have technical limitations. This report may still have undetectable risks, and the company will not bear any responsibility for the resulting losses.