FraVenner

Smart Contract Security Audit Report

FraVenner

Contents

FraVenner

1、executive summary

On July 24,2021,the FraVenner security team received the ART team's security audit application for ART, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

1.1 Project Introduction

Art chain is a fully decentralized community autonomous token based on BEP (BSC) network hairstyle. The token chips are dispersed in each user's hands through pre-sale. The token model is written in the smart contract and cannot be tampered with, which determines the manipulation of Wuzhuang and large households. It is a value token 100% circulating in the market and built by community consensus.

Contract Name:

ART Chain

Audit version file information:

Project source code

Initial audit files:

Address link:https://github.com/bbjs123/artContracts

Main Contract address:

0xe97364f13Fc8f40B0ab372a51b645a648f091483

1.2 Audit Results

Audit Result: Low risk, normal pass

Audit Number: 202107300028

Contract audit start date: July 24,2021

Contract audit completion date: July 30,2021

Audit Team: FraVenner Security Team

FraVenner

2、Audit Results

2.1 Audit type and results

Serial number	Audit type	Audit sub item	Audit results
1	Code specification audit	Compiler version security audit	Passed
		Audit of deprecated items	Passed
		Redundant code audit	Passed
		Safemath function audit	Passed
		Require / assert use audit	Passed
		Gas consumption audit	Passed
		Visibility specification audit	Passed
		The fallback function uses auditing	Passed
2	General vulnerability audit	Integer overflow audit	Passed
		Reentry attack audit	Passed
		Pseudo random number generation audit	Passed
		Transaction order dependent audit	Passed
		Denial of service attack audit	Passed
		Function call authority audit	Passed
		Call / delegatecall security audit	Passed
		Return value security audit	Passed
		Tx.origin uses security audit	Passed
		Replay attack audit	Passed
		Variable coverage audit	Passed
3	Business audit	Business logic audit	Passed
		Business realization audit	Passed

2.2 Description of audit results

FraVenner uses formal verification, static analysis, dynamic analysis, typical case testing and manual audit to conduct multi-dimensional and comprehensive security audit on the code standardization, security and business logic of the smart contract of art project. After audit, the smart contract of art project passed all test items, and the contract audit result was passed. The following is the detailed audit information of this contract.

2.2.1 Code specification audit

1. Compiler version security audit

The old version of the compiler may cause various known security problems. It is recommended that developers specify the contract code in the code and adopt the latest version Compiler version and eliminate compiler alarms.

safety suggestions: None

Audit results: Passed

2. Audit of deprecated items

The solid smart contract development language is in rapid iteration, and some keywords have been discarded by the new version of the compiler, such as throw Years, etc. in order to eliminate the possible hidden dangers, contract developers should not use keywords that have been discarded in the current compiler version.

safety suggestions: None

Audit results: Passed

3. Redundancy code audit

Redundant code in smart contract will reduce code readability and may consume more gas for contract deployment. It is recommended to eliminate it Redundant code.

safety suggestions: None

Audit results: Passed

4. Safemath function audit

Check whether the functions in the safemath library are correctly used in the contract for mathematical operations, or other anti overflow checks.

safety suggestions: None

Audit results: Passed

5. Require / assert use audit

Solid uses state recovery exceptions to handle errors. This mechanism will remove the form in the current call (and all its subcalls).State and mark the caller with an error. The functions assert and require can be used to check conditions and when conditions are not met.Throw an exception. The assert function can only be used to test for internal errors and check for non variables. The require function is used to confirm the validity of the condition,For example, whether the input variable or contract status variable meets the conditions, or verify the return value of external contract calls.

safety suggestions: None

Audit results: Passed

6. Gas consumption audit

The virtual machine needs gas to execute the contract code. When the gas is insufficient, the code execution will throw an outofgas exception and revoke the contract.There are status changes. Contract developers need to control the gas consumption of code to avoid the continuous failure of function execution due to insufficient gas.

safety suggestions: None

Audit results: Passed

7. Visibility specification audit

Check whether the visibility of contract functions meets the design requirements.

safety suggestions: None

Audit results: Passed

8. Audit of fallback function usage

Check whether the fallback function is used correctly in the current contract.

safety suggestions: None

Audit results: Passed

2.2.2 General vulnerability audit

1. Integer overflow audit

Integer overflow is a security problem in many languages, especially in smart contracts. Solidity can handle up to 256.A number of bits (2 * * 256-1). Increasing the maximum number by 1 will overflow and get 0. Similarly, when the number is uint type, 0 minus 1 will underflow.Maximum numeric value. Overflow conditions can lead to incorrect results, especially if their possible results are not expected, which may affect the performance of the program.Reliability and safety.

safety suggestions: None

Audit results: Passed

2. Reentry attack audit

Reentry vulnerability is the most typical heco smart contract vulnerability, which once led to the Dao being attacked. The cause of the vulnerability is in solidity.When the call. Value() function is used to send HT, it will consume all the gas it receives. When the call. Value() function is called to send HT.When the logical order of HT is wrong, there is a risk of reentry attack.

safety suggestions: None

Audit results: Passed

3. Pseudo random number generation audit

Random numbers may be used in smart contracts. In the case of solidity, block

information is often used as a random factor,However, such use is unsafe. Block information can be controlled by miners or obtained by attackers during trading. Such random numbers are in one.To some extent, it can be predicted or collided. A typical example is that the airdrop random number of fomo3d can be collided.

safety suggestions: None

Audit results: Passed

4. Transaction order dependent audit

During the packaging and execution of heco's transaction, when facing the transaction with the same difficulty, miners often choose the one with high gas cost as the priority.Package, so users can specify higher gas fees to make their transactions package and execute first.

safety suggestions: None

Audit results: Passed

5. Denial of service attack audit

Denial of service attack, or denial of service, can make the target unable to provide normal services. In the heco smart contract.There are such problems. Due to the immutability of smart contracts, such attacks may make the contracts never return to normal working state. Guide.There are many reasons for the denial of service of smart contracts, including malicious revert when being the transaction receiver and code design defects.Gas depletion, etc.

safety suggestions: None

Audit results: Passed

6. Function call authority audit

If the smart contract has high permission functions, such as coinage, self destruction, changeowner, etc., permission limits need to be made for function calls System to avoid security problems caused by authority disclosure.

safety suggestions: None

Audit results: Passed

7. Call / delegatecall security audit

Data is built internally in this contract (not arbitrarily specified by the user), and the corresponding call return value is checked, without security risk.

safety suggestions: None

Audit results: Passed

8. Return value security audit

There are transfer(), send(), call. Value() and other methods in the solidity. If the transfer fails, the transaction will be returned.Roll out, and the transfer failure of send and call.value will return. If the return is not judged correctly, it may be executed to the unpredicted state.Logic of period; In addition, in the implementation of the transfer / transferfrom function of hrc20token, it is also necessary to avoid the return of transfer failure.To avoid false recharge vulnerabilities.

Safety advice: None

Audit result: Passed

9. Tx.origin uses security audit

In the complex call of smart contract, tx.origin represents the address of the initial creator of the transaction. If tx.origin is used for transaction

Errors may occur in permission judgment; In addition, if the contract needs to judge whether the caller is a contract address, you need to use.Tx.origin, extcodesize cannot be used.

Safety advice: None

Audit result: Passed

10. Replay attack audit

Replay attack means that if two contracts use the same code implementation, and the authentication is in the transmission parameter, when the user is sending a message to one contract.When a transaction is executed in a contract, the transaction information can be copied and replayed to another contract to execute the transaction.

safety suggestions: None

Audit results: Passed

11. Variable coverage audit

It has complex variable types, such as structure, dynamic array, etc. if it is not used properly, its assignment may lead to.Overwriting the value of the existing state variable causes the contract execution logic exception.

safety suggestions: None

Audit results: Passed

2.2.3 Business audit

During the initial audit, the network security and information security team analyzed the project code manually combined with internal tools. During the audit, 46 problems were found, including 0 high-risk vulnerabilities, 39 medium risk vulnerabilities and 7 low-risk vulnerabilities. After communication, except for some negligible low-risk risks, the project party has rectified all other risks. The project risk is low and passed normally.

2.2.3.1 HIGH-risk vulnerabilities

There are no high-risk vulnerabilities in the projec.

2.2.3.2 Medium-risk vulnerabilities

1、Function could be marked as external.

(1) The function definition of "owner" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.461

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

Repair status: after communicating with the project party, the project party has repaired the above problems.

(2) The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.480

```
thereby removing any functionality that is only available to the owner.
*/
function renounceOwnership() public virtual onlyOwner {
emit OwnershipTransferred(_owner, address(0));
_owner = address(0);
}
/**
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(3) The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.489

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
require(newOwner != address(0), "Ownable: new owner is the zero address");
```

```
emit OwnershipTransferred(_owner, newOwner);
_owner = newOwner;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(4) The function definition of "setUsdtAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.582

```
event IntroLog(address indexed user,uint256 amount,uint256 time,address indexed intro);
function setOpenGetTime( uint256 _openGetTime) public onlyOwner{
openGetTime = _openGetTime;
}
function setUsdtAddress( address _usdtAddress) public onlyOwner{
usdtAddress = _usdtAddress;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(5) The function definition of "setUsdtAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.586

```
}
function setUsdtAddress( address _usdtAddress) public onlyOwner{
usdtAddress = _usdtAddress;
}
function setTokenAddress( address _tokenAddress) public onlyOwner{
tokenAddress = _tokenAddress;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(6) The function definition of "setTokenAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.589

```
usdtAddress = _usdtAddress;
}
function setTokenAddress( address _tokenAddress) public onlyOwner{
tokenAddress = _tokenAddress;
}
function setReceiveAddress( address _receiveAddress) public onlyOwner{
receiveAddress = _receiveAddress;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(7) The function definition of "setReceiveAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider tomark it as "external" instead.

Code location: art.sol.592

```
}
function setReceiveAddress( address _receiveAddress) public onlyOwner{
receiveAddress = _receiveAddress;
}
function setPrice( uint256 _price) public onlyOwner{
price = _price;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(8) The function definition of "setPrice" is marked "public". However, it is never directly called by another function in the same contract or in any of its

descendants. Consider to mark it as"external" instead.

Code location: art.sol.595

```
receiveAddress = _receiveAddress;
}
function setPrice( uint256 _price) public onlyOwner{
price = _price;
}
function setMinnum( uint256 _minnum) public onlyOwner{
minnum = _minnum;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(9) The function definition of "setMinnum" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.598

```
price = _price;
}
function setMinnum( uint256 _minnum) public onlyOwner{
minnum = _minnum;
}
function setMaxnum( uint256 _maxnum) public onlyOwner{
maxnum = _maxnum;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(10) The function definition of "setMaxnum" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark itas "external" instead.

Code location: art.sol.601

```
minnum = _minnum;
}
function setMaxnum( uint256 _maxnum) public onlyOwner{
maxnum = _maxnum;
}
function setRecRatio( uint256 _recRatio) public onlyOwner{
recRatio = _recRatio;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(11) The function definition of "setRecRatio" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.604

```
maxnum = _maxnum;
}
function setRecRatio( uint256 _recRatio) public onlyOwner{
recRatio = _recRatio;
}
/*
* 投资
*/
function invest (uint256 amount, address intro) public nonReentrant {
require(haveBuyAmount != totalAmount,'预售已经结束');
require(amount > 0,'购买数量必须大于 0');
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(12) The function definition of "invest" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as"external" instead.

FraVenner

Code location: art.sol.612

```
* 投资
*/
function invest (uint256 amount, address intro) public nonReentrant {
require(haveBuyAmount != totalAmount,'预售已经结束');
require(amount > 0,'购买数量必须大于 0');
require(amount >= minnum,'起步购买 1 份');
require(userCount[msg.sender].add(amount) <= maxnum, '不能超过最大限额');
require(haveBuyAmount.add(amount) <= totalAmount,'预售余额不足');
if(parents[msg.sender] == address(0) && intro != address(0)){
parents[msg.sender] = intro;
}
uint256 decimals = IERC20(usdtAddress).decimals();
uint usdtAmount = amount.mul(price).mul(10**decimals).div(10**18);
require(IERC20(usdtAddress).balanceOf(msg.sender) >= usdtAmount,'USDT 代币数量不足');
IERC20(usdtAddress).safeTransferFrom(msg.sender, receiveAddress, usdtAmount);
userCount[msg.sender] = userCount[msg.sender].add(amount);
//推荐认购奖励
if(parents[msg.sender] != address(0)){
introRewards[parents[msg.sender]]                                                    =
introRewards[parents[msg.sender]].add(amount.mul(recRatio).div(100));
emit IntroLog(msg.sender,amount.mul(recRatio).div(100),block.timestamp,parents[msg.sender]);
}
//已认购总量
haveBuyAmount = haveBuyAmount.add(amount);
//投资日志
emit InvestLog(msg.sender,amount,block.timestamp,intro);
}
```

```
function withdrawToken()public nonReentrant{
//require(haveBuyAmount == totalAmount,'预售尚未结束');
require(openGetTime<now,'尚未开放领取');
uint256 amount = userCount[msg.sender];
require(amount>0 , '余额不足');
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(13) The function definition of "withdrawToken" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: art.sol.642

```
//require(haveBuyAmount == totalAmount,'预售尚未结束');
require(openGetTime<now,'尚未开放领取');
uint256 amount = userCount[msg.sender];
require(amount>0 , '余额不足');
userCount[msg.sender] = 0;
IERC20(tokenAddress).safeTransfer(msg.sender, amount);
emit withdrawTokenLog(msg.sender,amount,block.timestamp);
}
function withdrawReward()public nonReentrant{
//require(haveBuyAmount == totalAmount,'预售尚未结束');
require(openGetTime<now,'尚未开放领取');
uint256 amount = introRewards[msg.sender];
require(amount>0 , '推荐认购奖励余额不足');
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(14) The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.433

```
* thereby removing any functionality that is only available to the owner.
*/
function renounceOwnership() public virtual onlyOwner {
emit OwnershipTransferred(_owner, address(0));
_owner = address(0);
}
/**
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(15) The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.442

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
require(newOwner != address(0), "Ownable: new owner is the zero address");
emit OwnershipTransferred(_owner, newOwner);
_owner = newOwner;
}
function geUnlockTime() public view returns (uint256) {
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(16) The function definition of "geUnlockTime" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.448

```
}
function geUnlockTime() public view returns (uint256) {
return _lockTime;
}
//Locks the contract for owner for the amount of time provided
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(17) The function definition of "lock" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.453

```
//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
_previousOwner = _owner;
_owner = address(0);
_lockTime = now + time;
emit OwnershipTransferred(_owner, address(0));
}
//Unlocks the contract for owner when _lockTime is exceeds
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(18) The function definition of "unlock" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.461

```
//Unlocks the contract for owner when _lockTime is exceeds
function unlock() public virtual {
require(_previousOwner == msg.sender, "You don't have permission to unlock");
```

```
require(now > _lockTime , "Contract is locked until 7 days");
emit OwnershipTransferred(_owner, _previousOwner);
_owner = _previousOwner;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(19) The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.768

```
}
function name() public view returns (string memory) {
return _name;
}
function symbol() public view returns (string memory) {
return _symbol;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(20) The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.772

```
}
function symbol() public view returns (string memory) {
return _symbol;
}
function decimals() public view returns (uint8) {
```

```
return _decimals;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(21) The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.776

```
}
function decimals() public view returns (uint8) {
return _decimals;
}
function totalSupply() public view override returns (uint256) {
return _tTotal;
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(22) The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.780

```
}
function totalSupply() public view override returns (uint256) {
return _tTotal;
}
function balanceOf(address account) public view override returns (uint256) {
if (_isExcluded[account]) return _tOwned[account];
```

```
return tokenFromReflection(_rOwned[account]);
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(23) The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.789

```
}
function transfer(address recipient, uint256 amount) public override returns (bool) {
_transfer(_msgSender(), recipient, amount);
return true;
}
function allowance(address owner, address spender) public view override returns (uint256) {
return _allowances[owner][spender];
}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(24) The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.794

```
}
function allowance(address owner, address spender) public view override returns (uint256) {
return _allowances[owner][spender];
}
function approve(address spender, uint256 amount) public override returns (bool) {
```

```
_approve(_msgSender(), spender, amount);
return true;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(25) The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.798

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

Repair status: after communicating with the project party, the project party has repaired the above problems.

(26) The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.803

```
}
function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
```

```
_transfer(sender, recipient, amount);
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
return true;
}
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(27) The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.809

```
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
return true;
}
function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(28) The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.814

```
}
function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
return true;
}
function isExcludedFromReward(address account) public view returns (bool) {
return _isExcluded[account];
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(29) The function definition of "isExcludedFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.819

```
}
function isExcludedFromReward(address account) public view returns (bool) {
return _isExcluded[account];
}
function totalFees() public view returns (uint256) {
return _tFeeTotal;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(30) The function definition of "totalFees" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.823

```
}
function totalFees() public view returns (uint256) {
return _tFeeTotal;
}
function deliver(uint256 tAmount) public {
address sender = _msgSender();
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(31) The function definition of "deliver" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.827

```
}
function deliver(uint256 tAmount) public {
address sender = _msgSender();
require(!_isExcluded[sender], "Excluded addresses cannot call this function");
(uint256 rAmount,,,,,) = _getValues(tAmount);
_rOwned[sender] = _rOwned[sender].sub(rAmount);
_rTotal = _rTotal.sub(rAmount);
_tFeeTotal = _tFeeTotal.add(tAmount);
}
function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view
returns(uint256) {
require(tAmount <= _tTotal, "Amount must be less than supply");
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(32)The function definition of "reflectionFromToken" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.836

```
}
function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view
returns(uint256) {
require(tAmount <= _tTotal, "Amount must be less than supply");
if (!deductTransferFee) {
(uint256 rAmount,,,,,) = _getValues(tAmount);
return rAmount;
} else {
(,uint256 rTransferAmount,,,,) = _getValues(tAmount);
return rTransferAmount;
}
}
function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
require(rAmount <= _rTotal, "Amount must be less than total reflections");
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(33) The function definition of "excludeFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.853

```
}
function excludeFromReward(address account) public onlyOwner() {
// require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap
router.');
require(!_isExcluded[account], "Account is already excluded");
if(_rOwned[account] > 0) {
_tOwned[account] = tokenFromReflection(_rOwned[account]);
```

```
}
_isExcluded[account] = true;
_excluded.push(account);
}
function includeInReward(address account) external onlyOwner() {
require(_isExcluded[account], "Account is already excluded");
for (uint256 i = 0; i < _excluded.length; i++) {
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(34) The function definition of "excludeFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.886

```
}
function excludeFromFee(address account) public onlyOwner {
_isExcludedFromFee[account] = true;
}
function includeInFee(address account) public onlyOwner {
_isExcludedFromFee[account] = false;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(35) The function definition of "includeInFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.890

```
}
function includeInFee(address account) public onlyOwner {
_isExcludedFromFee[account] = false;
```

```
}
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
_taxFee = taxFee;
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(36) The function definition of "setSwapAndLiquifyEnabled" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.920

```
}
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
swapAndLiquifyEnabled = _enabled;
emit SwapAndLiquifyEnabledUpdated(_enabled);
}
//to recieve ETH from uniswapV2Router when swaping
receive() external payable {}
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

(37) The function definition of "isExcludedFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Code location: arttoken.sol.1068

```
}
function isExcludedFromFee(address account) public view returns(bool) {
return _isExcludedFromFee[account];
}
function _approve(address owner, address spender, uint256 amount) private {
require(owner != address(0), "ERC20: approve from the zero address");
```

2、Read of persistent state following external call.

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Code location: arttoken.sol.763

```
//exclude owner and this contract from fee
_isExcludedFromFee[owner()] = true;
_isExcludedFromFee[address(this)] = true;
emit Transfer(address(0), _msgSender(), _tTotal);
```

Repair status: after communicating with the project party, the project party has repaired the above problems.

3、Write to persistent state following external call.

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Code location: arttoken.sol.765

```
//exclude owner and this contract from fee
_isExcludedFromFee[owner()] = true;
_isExcludedFromFee[address(this)] = true;
emit Transfer(address(0), _msgSender(), _tTotal);
}
```

Repair status: after communicating with the project party, the project party

has repaired the above problems.

2.2.3.3 LOW-risk vulnerabilities

1、A floating pragma is set.

(1) The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Code location: art.sol.2

```
// SPDX-License-Identifier: SimPL-2.0
pragma solidity ^0.6.0;
// import '@openzeppelin/contracts/access/Ownable.sol';
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

(2) The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Code location: art.sol.497

```
pragma solidity ^0.6.0;

/**
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

(3) The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Code location: arttoken.sol.3

```
pragma solidity ^0.6.12;
// SPDX-License-Identifier: Unlicensed
interface IERC20 {
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

2、Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Code location: art.sol.335

```
// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.call.value(weiValue)(data);
if (success) {
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

3、A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract.Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Code location: art.sol.755

```
IUniswapV2Router02 _uniswapV2Router
= IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
// Create a uniswap pair for this new token
```

```
uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
.createPair(address(this), _uniswapV2Router.WETH());
// set the rest of the contract variables
uniswapV2Router = _uniswapV2Router;
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

4、Multiple calls are executed in the same transaction.

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they’re part of your own codebase).

Code location: art.sol.758

```
.createPair(address(this), _uniswapV2Router.WETH());
// set the rest of the contract variables
uniswapV2Router = _uniswapV2Router;
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

5、Call with hardcoded gas amount.

The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.

Code location: art.sol.1186

```
payable(receiveAddress).transfer(address(this).balance);
}
}
//this method is responsible for taking all fee, if takeFee is true
function _tokenTransfer(address sender, address recipient, uint256 amount,bool takeFee) private
{
if(!takeFee)
```

Repair status: this vulnerability does not affect the overall security of the project, which has been ignored by the project party.

3、Statement

FraVenner issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these. For the facts that occurred or existed after the issuance, FraVenner is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to FraVenner by the information provider till the date of the insurance this report (referred to as "provided information").

FraVenner assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the FraVenner shall not be liable for any loss or adverse effect resulting therefrom. FraVenner only conducts the agreed security audit on the security situation of the project and issues this report. FraVenner is not responsible for the background and other conditions of the project.

FraVenner

Twitter: @fra_venner

FaceBook: @Fra Venner

E-mail: FraVenner@outlook.com