Smart Contract Security Audit Report

Contents

1、executive summary

On April 27, 2021, the Wanganxin security team received the MGS team's security audit application for MGS, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The Wanganxin security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

Wanganxin Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

Wanganxin Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2、Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack
- Timestamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3、Project Background

3.1 Project Information

Contract Name:

Magic Shouler (MGS)

Audit version file information:

Project source code

Initial audit files:

mgs-contracts.zip(SHA256):

8E8B064679986B367E43B474A4C74C1E19B5E10F59AB85B844CDD4CEC4AAF9A7

3.2 Project Structure

```
├─── StakingPool.sol
└─── Token.sol
```

4、Code Overview

4.1 Main Contract address

Not yet deployed on the mainnet.

4.2 Contracts Description

The Wanganxin Security team analyzed the visibility of major contracts during the audit, the result as follows:

Token			
Function Name	Visibility	Mutability	Modifiers
totalSupply	External	-	NO
balanceOf	External	-	NO
transfer	External	Can modify state	NO

allowance	External	-	NO
approve	External	Can modify state	NO
transferFrom	External	Can modify state	NO
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
div	Internal	-	-
mod	Internal	-	-
<Constructor>	Internal	Can modify state	Ownable
owner	Public	-	NO
renounceOwnership	Public	Can modify state	onlyOwner
transferOwnership	Public	Can modify state	onlyOwner
_setOwne	Internal	Can modify state	-
getSaleState	Public	-	NO
setupPubSale	Public	Can modify state	onlyOwner
buyToken	Public	Payable	NO
stopSale	Public	Can modify state	onlyOwner
doSale	Internal	Can modify state	-
doStopSale	Internal	Can modify state	-
<Constructor>	Public	Can modify state	PubSale
_addInitSupply	Interna	Can modify state	-
name	Public	-	NO
symbol	Public	-	NO
decimals	Public	-	NO
totalSupply	Public	-	NO
balanceOf	Public	-	NO
transfe	Public	Can modify state	NO
allowance	Public	-	NO
approve	Public	Can modify state	NO
transferFrom	Public	Can modify state	NO
increaseAllowance	Public	Can modify state	NO
decreaseAllowance	Public	Can modify state	NO
_transfer	Internal	Can modify state	-
_approve	Internal	Can modify state	-
_beforeTokenTransfer	Internal	-	-
_burn	Internal	Can modify state	-
burn	Public	Can modify state	NO

burnFrom	Public	Can modify state	NO
getSwapPair	Public	-	NO
setSwapPair	Public	Can modify state	onlyOwner
lockedBalance	Public	-	NO
availableBalance	Public	-	NO
getBalanceDetail	Public	-	NO
_addLock	Internal	Can modify state	-
linearLock	Public	Can modify state	NO
doSale	Internal	Can modify state	-
doStopSale	Internal	Can modify state	-
isContract	Internal	-	-
owner	Public	-	NO
max	Internal	-	-
min	Internal	-	-
average	Internal	-	-
isContract	Internal	-	-
sendValue	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionStaticCall	Internal	-	-
_verifyCallResult	Private	-	-
safeTransfe	Internal	Can modify state	-
safeTransferFrom	Internal	Can modify state	-
safeApprove	Internal	Can modify state	-
safeIncreaseAllowance	Internal	Can modify state	-
safeDecreaseAllowance	Internal	Can modify state	-
_callOptionalReturn	Private	Can modify state	-
lastTimeRewardApplicable	Public	-	NO
rewardPerToken	Public	-	NO
earned	Public	-	NO
getRewardForDuration	External	-	NO
stake	External	Can modify state	nonReentrant
withdraw	Public	Can modify state	nonReentrant
getReward	Public	Can modify state	nonReentrant
exit	External	Can modify state	NO
setupPool	External	Can modify state	nonReentrant

getPoolView	Public	-	NO

Staking			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Internal	Can modify state	-
owner	Public	-	-
renounceOwnership	Public	Can modify state	onlyOwner
transferOwnership	Public	Can modify state	onlyOwner
_setOwner	Internal	Can modify state	-
max	Internal	-	-
min	Internal	-	-
average	Internal	-	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
div	Internal	-	-
mod	Internal	-	-
totalSupply	External	-	NO
balanceOf	External	-	NO
transfer	External	Can modify state	NO
allowance	External	-	NO
approve	External	Can modify state	NO
transferFrom	External	Can modify state	NO
isContrac	Internal	-	-
sendValue	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionStaticCall	Internal	-	-
_verifyCallResult	Private	-	-
safeTransfer	Internal	Can modify state	-
safeTransferFrom	Internal	Can modify state	-
safeApprove	Internal	Can modify state	-
safeIncreaseAllowance	Internal	Can modify state	-
safeDecreaseAllowance	Internal	Can modify state	-
_callOptionalReturn	Private	Can modify state	-
<Constructor>	Public	Can modify state	Ownable

lastTimeRewardApplicable	Public	-	NO
rewardPerToken	Public	-	NO
earned	Public	-	NO
getRewardForDuration	External	-	NO
stake	External	Can modify state	nonReentrant
withdraw	Public	Can modify state	nonReentrant
getReward	Public	Can modify state	nonReentrant
exit	External	Can modify state	-
setupPool	External	Can modify state	nonReentrant
getPoolView	Public	-	-
getSaleState	Public	-	NO
setupPubSale	Public	Can modify state	onlyOwner
buyToken	Public	Payable	NO
stopSale	Public	Can modify state	onlyOwner
doSale	Internal	Can modify state	-
doStopSale	Internal	Can modify state	-
<Constructor>	Public	Can modify state	PubSale
_addInitSupply	Internal	Can modify state	-
name	Public	-	NO
symbol	Public	-	NO
decimals	Public	-	NO
increaseAllowance	Public	Can modify state	NO
decreaseAllowance	Public	Can modify state	NO
_transfe	Internal	Can modify state	-
_approve	Internal	Can modify state	-
_beforeTokenTransfer	Internal	-	-
_burn	Internal	Can modify state	-
burn	Public	Can modify state	NO
burnFrom	Public	Can modify state	NO
getSwapPair	Public	-	NO
setSwapPair	Public	Can modify state	onlyOwner
lockedBalance	Public	-	NO
availableBalance	Public	-	NO
getBalanceDetail	Public	-	NO
_addLock	Internal	Can modify state	-
linearLock	Public	Can modify state	NO

4.3 Code Audit

4.3.1 Medium-risk vulnerabilities

4.3.1.1 Function could be marked as external

The function definition of "linearLock" 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.

1. Code location:token.sol.291

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

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

2. Code location:token.sol.310

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

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

3. Code location:token.sol.319

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
```

```
_setOwner(newOwner);
}

function _setOwner(address newOwner) internal {
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

4. Code location:token.sol.358

```
 SaleState private _saleState;

function getSaleState() public view returns (SaleState memory) {
return _saleState;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

5. Code location:token.sol.371

```
constructor(address aOwner) internal Ownable(aOwner) {}

function setupPubSale(
address payee,
uint256 rate,
uint256 cap,
uint256 lockDuration,
uint256 beginTime,
uint256 endTime
) public onlyOwner {
require(
block.timestamp < beginTime && beginTime < endTime,
"Invalid time"
);
require(payee != address(0), "Payee is zero");
require(rate > 0, "Rate is zero");
require(
_saleState.beginTime == 0 || block.timestamp < _saleState.beginTime,
```

```
"Setup is not allowed anymore"
);
_saleState.payee = payee;
_saleState.rate = rate;
_saleState.cap = cap;
_saleState.lockDuration = lockDuration;
_saleState.beginTime = beginTime;
_saleState.endTime = endTime;
_saleState.remain = cap;
emit SetupSale(beginTime, endTime, cap, rate);
}

function buyToken() public payable {
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

6. Code location: token.sol.399

```
 }

function buyToken() public payable {
require(
block.timestamp >= _saleState.beginTime &&
block.timestamp < _saleState.endTime,
"Not the right time"
);
require(_saleState.remain > 0, "No more token");
require(msg.value > 0, "Value is 0");
uint256 amount = msg.value.mul(_saleState.rate);
uint256 refund = 0;
if (amount > _saleState.remain) {
amount = _saleState.remain;
refund = msg.value.sub(amount.div(_saleState.rate));
if (refund == msg.value) {
refund = refund.sub(1);
}
}
```

```
_saleState.totalSold = _saleState.totalSold.add(amount);
_saleState.remain = _saleState.remain.sub(amount);
payable(_saleState.payee).transfer(msg.value.sub(refund));
doSale(msg.sender, amount, _saleState.lockDuration);
if (refund > 0) {
msg.sender.transfer(refund);
}
emit BuyToken(msg.sender, amount);
}

function stopSale() public onlyOwner {
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

7. Code location:token.sol.426

```
 }

function stopSale() public onlyOwner {
require(_saleState.remain > 0, "Invalid state");
emit StopSale(_saleState.remain);
doStopSale(_saleState.remain);
_saleState.remain = 0;
}

function doSale(
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

8. Code location:token.sol.530

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

/**
```

Fix status: This issues has been ignore.

9. **Code location:**token.sol.538

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

/**
```

Fix status: This issues has been ignore.

10. **Code location:**token.sol.551

```
 * {IERC20-balanceOf} and {IERC20-transfer}.
*/
function decimals() public view returns (uint8) {
return _decimals;
}

/**
```

Fix status: This issues has been ignore.

11. **Code location:**token.sol.558

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

/**
```

Fix status: This issues has been ignore.

12. **Code location:**token.sol.565

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

/**
```

Fix status: This issues has been ignore.

13. Code location:token.sol.577

```
* - the caller must have a balance of at least `amount`.
*/
function transfer(address recipient, uint256 amount)
public
virtual
override
returns (bool)
{
// burn on buy or removeLiquidity from swap
uint256 buyBurnLimit = _pairs[msg.sender].buyBurnLimit;
bool triggerBuyBurn =
buyBurnLimit > 0 && _balances[msg.sender] > buyBurnLimit;

_transfer(msg.sender, recipient, amount);
uint256 buyBurnPercent = _pairs[msg.sender].buyBurnPercent;
if (triggerBuyBurn && buyBurnPercent > 0) {
_burn(recipient, amount.mul(buyBurnPercent).div(100));
}
return true;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

14. Code location:token.sol.616

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount)
public
virtual
override
returns (bool)
{
```

```
_approve(msg.sender, spender, amount);
return true;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

15. Code location:token.sol.639

```
* `amount`.
*/
function transferFrom(
address sender,
address recipient,
uint256 amount
) public virtual override returns (bool) {
_transfer(sender, recipient, amount);
_approve(
sender,
msg.sender,
_allowances[sender][msg.sender].sub(
amount,
"ERC20: transfer amount exceeds allowance"
)
);
// burn on sell or addLiquidity to swap
uint256 sellBurnLimit = _pairs[recipient].sellBurnLimit;
uint256 sellBurnPercent = _pairs[recipient].sellBurnPercent;
if (
sellBurnLimit > 0 &&
sellBurnPercent > 0 &&
_balances[recipient] > sellBurnLimit
) {
_burn(recipient, amount.mul(sellBurnPercent).div(100));
}
return true;
```

```
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

16. Code location:token.sol.678

```
* - `spender` cannot be the zero address.
*/
function increaseAllowance(address spender, uint256 addedValue)
public
virtual
returns (bool)
{
_approve(
msg.sender,
spender,
_allowances[msg.sender][spender].add(addedValue)
);
return true;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

17. Code location:token.sol.705

```
* `subtractedValue`.
*/
function decreaseAllowance(address spender, uint256 subtractedValue)
public
virtual
returns (bool)
{
_approve(
msg.sender,
spender,
```

```
_allowances[msg.sender][spender].sub(
subtractedValue,
"ERC20: decreased allowance below zero"
)
);
return true;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

18. Code location:token.sol.835

```
* See {ERC20-_burn}.
*/
function burn(uint256 amount) public virtual {
_burn(msg.sender, amount);
}

/**
```

19. Code location:token.sol.850

```
* `amount`.
*/
function burnFrom(address account, uint256 amount) public virtual {
uint256 decreasedAllowance =
allowance(account, msg.sender).sub(
amount,
"ERC20: burn amount exceeds allowance"
);

_approve(account, msg.sender, decreasedAllowance);
_burn(account, amount);
}

// ==================== Burn on swap ====================
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

20. Code location:token.sol.862

```
// ==================== Burn on swap ====================
function getSwapPair(address pair) public view returns (SwapPair memory) {
return _pairs[pair];
}

function setSwapPair(
```

Fix status: This issues has been ignore.

21. Code location:token.sol.866

```
}

function setSwapPair(
address pair,
uint256 buyBurnLimit,
uint256 buyBurnPercent,
uint256 sellBurnLimit,
uint256 sellBurnPercent
) public onlyOwner {
require(
buyBurnPercent <= 100 && sellBurnPercent <= 100,
"Invalid percent number"
);
_pairs[pair] = SwapPair({
buyBurnLimit: buyBurnLimit,
buyBurnPercent: buyBurnPercent,
sellBurnLimit: sellBurnLimit,
sellBurnPercent: sellBurnPercent
});
}

// ==================== Locking ====================
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

22. Code location:token.sol.902

```
}

function getBalanceDetail(address account)
public
view
returns (BalanceDetail memory)
{
LockInfo memory li = _locks[account];
return
BalanceDetail({
balance: _balances[account],
available: availableBalance(account),
lockAmount: li.amount,
lockDuration: li.duration,
lockDeadline: li.deadline
});
}

function _addLock(
```

Fix status: This issues has been ignore.

23. Code location:token.sol.944

```
* The new lock-deadline time must not be earlier than the previous lock.
*/
function linearLock(uint256 amount, uint256 duration) public {
_addLock(msg.sender, amount, duration);
}

// ==================== PubSale ====================
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

24. Code location:token.sol.36

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

```
return _owner;
}

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

25. Code location:token.sol.55

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

/**
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

26. Code location:token.sol.64

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
_setOwner(newOwner);
}

function _setOwner(address newOwner) internal {
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

27. Code location:token.sol.803

```
 }

function getPoolView() public view returns (PoolView memory pv) {
pv.rewardsToken = rewardsToken;
pv.stakingToken = stakingToken;
pv.periodFinish = periodFinish;
pv.rewardRate = rewardRate;
```

```
pv.rewardsDuration = rewardsDuration;
pv.lastUpdateTime = lastUpdateTime;
pv.rewardPerTokenStored = rewardPerTokenStored;
pv.totalSupply = _totalSupply;
}
}
```

Fix status: After communicating with the project party, the project side has fixed the above problems.

4.3.2 Low-risk vulnerabilities

4.3.2.1 Unused function parameter "buyer"

Code location: token.sol.434

```
function doSale(
address buyer,
uint256 amount,
uint256 duration
```

Fix status: This issues has been ignore.

4.3.2.2 Unused function parameter “amount”

Code location: token.sol.435

```
function doSale(
address buyer,
uint256 amount,
uint256 duration
) internal virtual {}
```

Fix status: This issues has been ignore.

4.3.2.3 Unused function parameter “ duration”

Code location: token.sol.436

```
address buyer,
uint256 amount,
uint256 duration
) internal virtual {}
```

Fix status: This issues has been ignore.

4.3.2.4 Unused function parameter “ duration”

Code location:token.sol.436

```
address buyer,
uint256 amount,
uint256 duration
) internal virtual {}
```

Fix status: This issues has been ignore.

4.3.2.5 Unused function parameter “ remain”

Code location:token.sol.439

```
) internal virtual {}

function doStopSale(uint256 remain) internal virtual {}
}
```

Fix status: This issues has been ignore.

4.3.2.6 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.

1. **Code location:**token.sol.418

```
_saleState.totalSold = _saleState.totalSold.add(amount);
_saleState.remain = _saleState.remain.sub(amount);
payable(_saleState.payee).transfer(msg.value.sub(refund));
```

```
doSale(msg.sender, amount, _saleState.lockDuration);
if (refund > 0) {
```

Fix status: This issues has been ignore.

2. Code location:token.sol.421

```
doSale(msg.sender, amount, _saleState.lockDuration);
if (refund > 0) {
msg.sender.transfer(refund);
}
emit BuyToken(msg.sender, amount);
```

Fix status: This issues has been ignore.

5、Audit Result

5.1 Conclusion

Audit Result: Low risk

Audit Number: 202104300128

Audit Date: April 30, 2021

Audit Team: Wanganxin Security Team

Summary conclusion: Initial audit,The Wanganxin security team use a manual and Wanganxin Team's analysis tool audit of the codes for security issues. There are 32 issues found during the audit. There are 27 medium-risk vulnerabilities and 5 low-risk vulnerabilities. After communication, except for some negligible risks, the project side has rectified all other risks. The known problems of the project have been rectified and the risk is low.

6、Statement

Wanganxin 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, Wanganxin 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 Wanganxin by the information provider till the date of the insurance this report (referred to as "provided information"). Wanganxin 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 Wanganxin shall not be liable for any loss or adverse effect resulting therefrom. Wanganxin only conducts the agreed security audit on the security situation of the project and issues this report. Wanganxin is not responsible for the background and other conditions of the project.

Website

https://www.wanganxin.com

E-mail

info@wanganxin.com