Smart Contract Security Audit

Contents

1. Executive Summary

On August 25, 2020, the SlowMist security team received the OneSwap team's security audit application for OneSwap, 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 SlowMist 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.

SlowMist 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 wether there are vulnerabilities in programs suck as nodes, SDK, etc.

SlowMist 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	Medium vulnerability will affect the operation of DeFi project. It is recommended

vulnerablities	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 Introduction

OneSwap is a fully decentralized exchange protocol on smart contract, with permission-free token listing and automated market making.

Project website:

https://www.oneswap.net/

Audit version code:

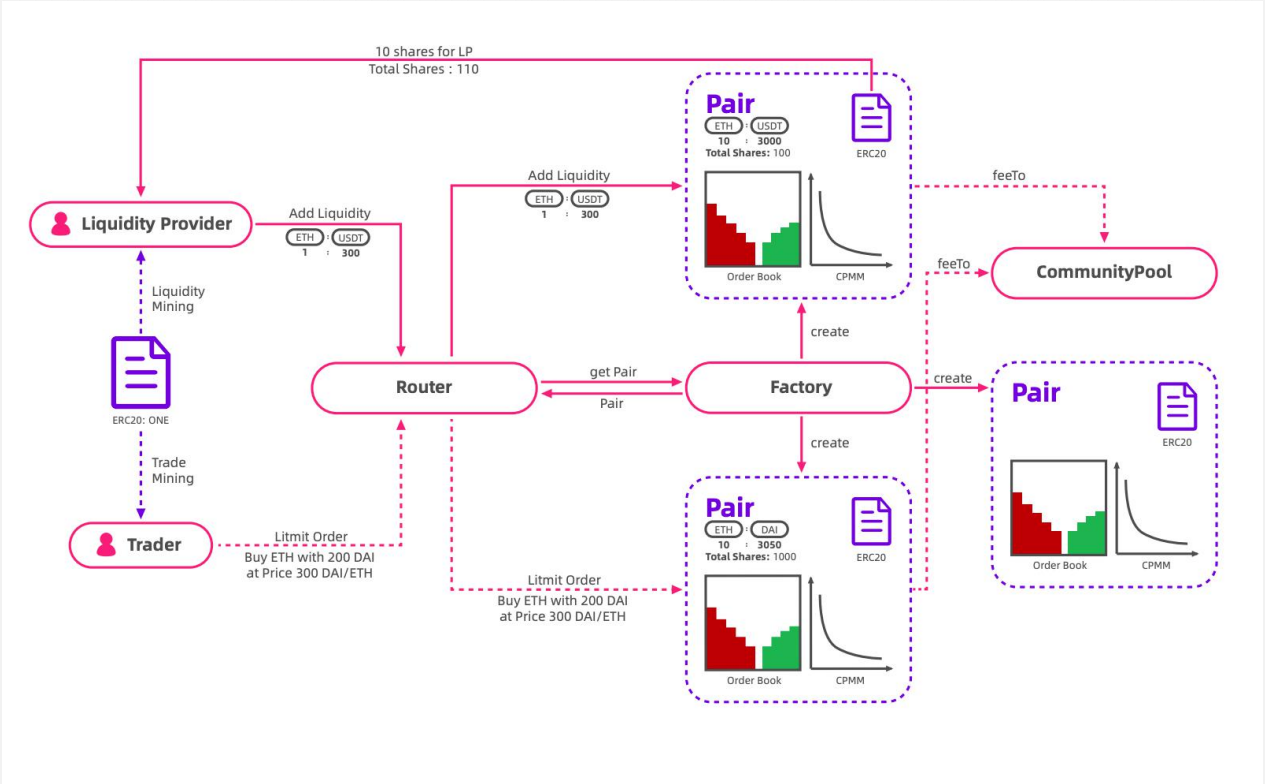
https://github.com/oneswap/oneswap_contract_ethereum/tree/300261dabed260bf48cdffcc96f3c3c0293c5511/contracts

The documents provided by the OneSwap team are as follows:

OneSwap_WhitePaper_v1.0：

https://www.oneswap.net/whitepaper/OneSwap_WhitePaper_v1.0_en.pdf

Architecture diagram：



4. Code Overview

4.1 Contracts Description

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

OneSwapBlackList			
Function Name	Visibility	Mutability	Modifiers
owner	Public	-	-
newOwner	Public	-	-
isBlackListed	Public	-	-
changeOwner	Public	Can modify state	onlyOwner
updateOwner	Public	Can modify state	onlyNewOwner
addBlackLists	Public	Can modify state	onlyOwner
removeBlackLists	Public	Can modify state	onlyOwner
_setOwner	Internal	Can modify state	-

LockSend			
Function Name	Visibility	Mutability	Modifiers
lockSend	Public	Can modify state	beforeUnlockTime
unlock	Public	Can modify state	afterUnlockTime
_getLockedSendKey	Private	-	-
_safeTransferToMe	Internal	Can modify state	-
_safeTransfer	Internal	Can modify state	-

OneSwapBuyback			
Function Name	Visibility	Mutability	Modifiers
addMainToken	External	Can modify state	-
removeMainToken	External	Can modify state	-
isMainToken	External	-	-
mainTokens	External	-	-
removeLiquidity	External	Can modify state	-
_removeLiquidity	Private	Can modify state	-
swapForMainToken	External	Can modify state	-
_swapForMainToken	Private	Can modify state	-
swapForOnesAndBurn	External	Can modify state	-

_swapForOnes	Private	Can modify state	-
receive()	External	Payable	-

OneSwapFactory			
Function Name	Visibility	Mutability	Modifiers
createPair	External	Can modify state	-
_getDecimals	Private	-	-
allPairsLength	External	-	-
setFeeTo	External	Can modify state	-
setFeeToSetter	External	Can modify state	-
setPairLogic	External	Can modify state	-
setFeeBPS	External	Can modify state	-
getTokensFromPair	External	-	-
tokensToPair	External	-	-

OneSwapGov			
Function Name	Visibility	Mutability	Modifiers
proposalInfo	External	-	-
voterInfo	External	-	-
submitFundsProposal	External	Can modify state	-
submitParamProposal	External	Can modify state	-
submitUpgradeProposal	External	Can modify state	-
submitTextProposal	External	Can modify state	-
_newProposal	Private	Can modify state	-
vote	External	Can modify state	-
_vote	Private	Can modify state	-
_getVoterInfo	Private	-	-
_setVoterInfo	Private	Can modify state	-
tally	External	Can modify state	-
_resetProposal	Private	Can modify state	-

_execProposal	Private	Can modify state	-
_taxProposer	Private	Can modify state	-
withdrawOnes	External	Can modify state	-

OneSwapPair			
Function Name	Visibility	Mutability	Modifiers
_expandPrice	Private	-	-
symbol	External	-	-
_emitNewLimitOrder	Private	-	-
_emitNewMarketOrder	Private	-	-
_emitOrderChanged	Private	-	-
_emitDealWithPool	Private	-	-
_emitRemoveOrder	Private	-	-
_order2uint	Internal	-	-
_uint2order	Internal	-	-
_hasOrder	Internal	-	-
_getOrder	Internal	-	-
_setOrder	Internal	Can modify state	-
_deleteOrder	Internal	Can modify state	-
_getFirstOrderID	Internal	-	-
_setFirstOrderID	Internal	Can modify state	-
removeOrders	External	Can modify state	lock
removeOrder	External	Can modify state	lock
_removeOrder	Private	Can modify state	-
_removeOrderFromBook	Internal	Can modify state	-
_insertOrderAtHead	Private	Can modify state	-
_getOrder3Times	Private	-	-
_insertOrderFromGivenPos	Private	Can modify state	-
_insertOrderFromHead	Private	Can modify state	-
_insertOrder	Private	Can modify state	-
getPrices	External	Can modify state	-
getOrderList	External	-	-
_getUnusedOrderID	Internal	-	-
calcStockAndMoney	External	-	-
_calcStockAndMoney	Private	-	-

addLimitOrder	External	Payable	-
addMarketOrder	External	Payable	-
_checkRemainAmount	Private		-
_addOrder	Private	Can modify state	-
_intopoolAmountTillPrice	Private	-	-
_tryDealInPool	Private	-	-
_dealInOrderBook	Internal	Can modify state	-
_dealWithPoolAndCollectFee	Internal	Can modify state	-
_insertOrderToBook	Internal	Can modify state	-

OneSwapRouter			
Function Name	Visibility	Mutability	Modifiers
_addLiquidity	Private	-	-
addLiquidity	External	Payable	ensure
_removeLiquidity	Private	-	-
removeLiquidity	External	Can modify state	ensure
_swap	Internal	Can modify state	-
swapToken	External	Payable	ensure
limitOrder	External	Payable	ensure
_safeTransferFrom	Internal	Can modify state	-
_safeTransferETH	Internal	Can modify state	-
_quote	Internal	-	-
_getTokensFromPair	Internal	-	-

OneSwapToken			
Function Name	Visibility	Mutability	Modifiers
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-

totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can modify state	-
allowance	Public	-	-
approve	Public	Can modify state	-
transferFrom	Public	Can modify state	-
increaseAllowance	Public	Can modify state	-
decreaseAllowance	Public	Can modify state	-
burn	Public	Can modify state	-
burnFrom	Public	Can modify state	-
multiTransfer	Public	Can modify state	-
_transfer	Internal	Can modify state	-
_burn	Internal	Can modify state	-
_approve	Internal	Can modify state	-
_beforeTokenTransfer	Internal	-	-

SupervisedSend			
Function Name	Visibility	Mutability	Modifiers
supervisedSend	Public	Can modify state	beforeUnlockTime
supervisedUnlockSend	Public	Can modify state	afterUnlockTime
earlyUnlockBySupervisor	Public	Can modify state	beforeUnlockTime
earlyUnlockBySender	Public	Can modify state	beforeUnlockTime
_getSupervisedSendKey	Private	-	-
_safeTransferToMe	Internal	Can modify state	-
_safeTransfer	Internal	Can modify state	-

4.2 Gas Consumption

Using the code of these two projects for a gas consumption test:

https://github.com/oneswap/oneswap_contract_ethereum
https://github.com/oneswap/Uniswap-gas-test

The OneSwap Development Team optimized gas consumption for users as much as possible in the project, which greatly reduced gas consumption.

OneSwap Gas Consumption

Operation	Gas Consumption
Create + add liquidity	419493
Add liquidity only	116409
Remove liquidity	97837
Swap once (only deal with pool, no counterparty orders)	125115
Place the first sell order erc20/ETH	131902
Remove orders from the order book	31589
Buy order partially deal, erc20/ETH trading pair	163502
Buy order completely deal, erc20/ETH trading pair	112836

Uniswap Gas Consumption

Operation	Gas Consumption
Create + add liquidity	2174541
Add liquidity only	123702
Remove liquidity	175966
Swap once	117503

4.3 Code Audit

4.3.1 Enhancement Suggestions

This is a enhancement suggestion, If stock contact or money contact has a issue, and the issue can lead to a contract cannot call normally, so cannot remove liquidity to get back the tokens of another contract, if the issue lead to a contact can't call forever, then the tokens of another contract will be locked forever too.

It is suggested to add voting logic to the governance contract. In case of such issue, can change the

logic of pair contract retrieving liquidity by voting to avoid the tokens of normal contract be locked.

contracts/OneSwapPair.sol

```
    function burn(address to) external override lock returns (uint stockAmount, uint moneyAmount) {
        uint[5] memory proxyData;
        ProxyData.fill(proxyData, 4+32*(ProxyData.COUNT+1));
        (uint112 reserveStock, uint112 reserveMoney, uint32 firstSellID) = getReserves();
        (uint bookedStock, uint bookedMoney, ) = getBooked();
        uint stockBalance = _myBalance(ProxyData.stock(proxyData)).sub(bookedStock);
        uint moneyBalance = _myBalance(ProxyData.money(proxyData)).sub(bookedMoney);
        require(stockBalance >= uint(reserveStock) && moneyBalance >= uint(reserveMoney), "OneSwap:
INVALID_BALANCE");

        bool feeOn = _mintFee(reserveStock, reserveMoney, proxyData);
        {
            uint _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can
update in _mintFee
            uint liquidity = balanceOf[address(this)]; // we're sure liquidity < totalSupply
            stockAmount = liquidity.mul(stockBalance) / _totalSupply;
            moneyAmount = liquidity.mul(moneyBalance) / _totalSupply;
            require(stockAmount > 0 && moneyAmount > 0, "OneSwap: INSUFFICIENT_BURNED");

            //_burn(address(this), liquidity);
            balanceOf[address(this)] = 0;
            totalSupply = totalSupply.sub(liquidity);
            emit Transfer(address(this), address(0), liquidity);
        }

        address ones = ProxyData.ones(proxyData);
        _safeTransfer(ProxyData.stock(proxyData), to, stockAmount, ones);
        _safeTransfer(ProxyData.money(proxyData), to, moneyAmount, ones);

        stockBalance = stockBalance - stockAmount;
        moneyBalance = moneyBalance - moneyAmount;

        _setReserves(stockBalance, moneyBalance, firstSellID);
        if (feeOn) _kLast = stockBalance.mul(moneyBalance);
        emit Burn(msg.sender, (moneyAmount<<112)|stockAmount, to);
    }
```

contracts/OneSwapPair.sol

```
function _safeTransfer(address token, address to, uint value, address ones) internal {
        if(token==address(0)) {
            // limit gas to 9000 to prevent gastoken attacks
            // solhint-disable-next-line avoid-low-level-calls
            to.call{value: value, gas: 9000}(new bytes(0)); //we ignore its return value purposely
            return;
        }
        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory data) = token.call(abi.encodeWithSelector(_SELECTOR, to, value));
        success = success && (data.length == 0 || abi.decode(data, (bool)));
        if(!success) { // for failsafe
            address onesOwner = IOneSwapToken(ones).owner();
            // solhint-disable-next-line avoid-low-level-calls
            (success, data) = token.call(abi.encodeWithSelector(_SELECTOR, onesOwner, value));
            require(success && (data.length == 0 || abi.decode(data, (bool))), "OneSwap:
TRANSFER_FAILED");
        }
    }
```

5. Audit Result

5.1 Conclusion

Audit Result : Passed

Audit Number : 0X002009040003

Audit Date : September 04, 2020

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist Team in-house analysis tool audit of the codes for security issues, no critical, high-risk, medium-risk or low-risk vulnerabilities were found during the audit. There is an enhancement suggestion.

6. Statement

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

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account