Smart contract security audit report

Powerswap smart contract security audit report

Audit Team : Noneage security team

Audit Date : March 31 , 2021

Powerswap Smart Contract Security Audit Report

1. Overview

On February 24, 2021, the security team of Noneage Technology received the security audit request of the **Powerswap project**. The team will conduct a report on the **Powerswap smart contract** from March 22, 2021 to March 31, 2021. During the audit process, the security audit experts of Noneage Technology communicate with the relevant interface people of the Powerswap project, maintain information symmetry, conduct security audits under controllable operational risks, and try to avoid project generation and operation during the test process. Cause risks.

Through communicat and feedback with Powerswap project party, it is confirmed that the loopholes and risks found in the audit process have been repaired or within the acceptable range. The result of this Powerswap smart contract security audit: **passed.**

Audit Report MD5：702DAD0C2480B9D2469C7F7ED69FE5F5

2. Background

2.1 Project Description

> Project name: Powerswap
>
> official website: https://ht.powerswap.org/
>
> Contract type: DeFi Token contract
>
> Code language: Solidity
>
> Contract documents: PowerswapFactory.sol、PowerswapFactory.sol、WETH9.sol

2.2 Audit Range

Powerswap officially provides the contract address on the chain:

PowerswapFactory https://cn.etherscan.com/address/0xB89658d9636744D0b016b4AC0d71935d667c2065#code

PowerswapRouter https://cn.etherscan.com/address/0xe03B41ccB844C277616e5AB7F25eCfB99Ad99917#code

WETH9 https://cn.etherscan.com/address/0x31Ea6627C958531823D50024028dF6635DD3faE5#code

2.3 Security Audit List

The security experts of Noneage Technology conduct security audits on the security audit list within the agreement, The scope of this smart contract security audit does not include new attack methods that may appear in the future, does not include the code after contract upgrades or tampering, and is not included in the subsequent cross-country, does not include cross-chain deployment, does not include project front-end code security and project platform server security.

This smart contract security audit list includes the following:

- Integer overflow
- Reentry attack
- Floating point numbers and numerical precision
- Default visibility
- Tx.origin authentication
- Wrong constructor
- Return value not verified
- Insecure random numbers
- Timestamp dependency
- Transaction order is dependent
- Delegatecall
- Call
- Denial of service
- Logic design flaws
- Fake recharge vulnerability
- Short address attack
- Uninitialized storage pointer
- Additional token issuance
- Frozen account bypass
- Access control
- Gas usage

3. Contract Structure Analysis

3.1 Directory Structure

```
|
└─Powerswap
    PowerswapFactory.sol
    PowerswapRouter.sol
    WETH9.sol
```

3.2 Powerswap contract

Contract

PowerswapERC20

- _mint(address to, uint value)

- _burn(address from, uint value)
- _approve(address owner, address spender, uint value)
- _transfer(address from, address to, uint value)
- approve(address spender, uint value)
- transfer(address to, uint value)
- transferFrom(address from, address to, uint value)
- permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s)

PowerswapPair

- getReserves()
- getBalances()
- getPrices()
- getR()
- _safeTransfer(address token, address to, uint value)
- initialize(address _token0, address _token1)
- setR(uint16 _R)
- _updateAtTheInitialBlock(uint balance0, uint balance1)
- _updateAtTheFirstCallInEachBlock(uint112 _reserve0, uint112 _reserve1, uint32 timeElapsed)
- _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1)
- _updateAfterSwap
- _mintFee(uint112 _reserve0, uint112 _reserve1)
- mint(address to)
- burn(address to)
- swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)
- updateVirtualBalances()
- skim(address to)
- sync()

PowerswapFactory

- allPairsLength()
- createPair(address tokenA, address tokenB)
- setFeeTo(address _feeTo)
- setFeeToSetter(address _feeToSetter)
- setrSetter(address _rSetter)

PowerswapRouter

- _addLiquidity(address tokenA,address tokenB,uint amountADesired,uint amountBDesired,uint amountAmin,uint amountBmin)
- addLiquidity(address tokenA,address tokenB,uint amountADesired,uint amountBDesired,uint amountAmin,uint amountBmin,address to,uint deadline)
- addLiquidityETH(address token,uint amountTokenDesired,uint amountTokenMin,uint amountETHMin,address to,uint deadline)

- removeLiquidity(address tokenA,address tokenB,uint liquidity,uint amountAMin,uint amountBMin,address to,uint deadline)
- removeLiquidityETH(address token,uint liquidity,uint amountTokenMin, uint amountETHMin,address to,uint deadline)
- _removeLiquidityWithPermit(address tokenA,address tokenB,uint liquidity,uint amountAMin, uint amountBMin,address to,uint deadline,bool approveMax, uint8 v, bytes32 r, bytes32 s)
- removeLiquidityWithPermit(address tokenA,address tokenB,uint liquidity,uint amountAMin,uint amountBMin,address to,uint deadline,bool approveMax, uint8 v, bytes32 r, bytes32 s)
- removeLiquidityETHWithPermit(address token,uint liquidity,uint amountTokenMin,uint amountETHMin,address to,uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s)
- removeLiquidityETHSupportingFeeOnTransferTokens(address token,uint liquidity,uint amountTokenMin,uint amountETHMin,address to,uint deadline)
- removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address token,uint liquidity,uint amountTokenMin,uint amountETHMin, address to, uint deadline,bool approveMax, uint8 v, bytes32 r, bytes32 s)
- function _swap(uint[] memory amounts, address[] memory path, address _to)
- function swapExactTokensForTokens(uint amountIn,uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapTokensForExactTokens(uint amountOut,uint amountInMax,address[] calldata path,address to,uint deadline)
- swapExactETHForTokens(uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline)
- swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline)
- swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline)
- _swapSupportingFeeOnTransferTokens(address[] memory path, address _to)
- swapExactTokensForTokensSupportingFeeOnTransferTokens(uint amountIn,uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapExactETHForTokensSupportingFeeOnTransferTokens(uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapExactTokensForETHSupportingFeeOnTransferTokens(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint deadline)
- quote(uint amountA, uint reserveA, uint reserveB)
- getAmountOut(uint amountIn, uint reserveIn, uint reserveOut, uint price, uint16 R)
- getAmountIn(uint amountOut, uint reserveIn, uint reserveOut, uint price, uint16 R)
- getAmountsOut(uint amountIn, address[] memory path)
- getAmountsIn(uint amountOut, address[] memory path)

WETH9

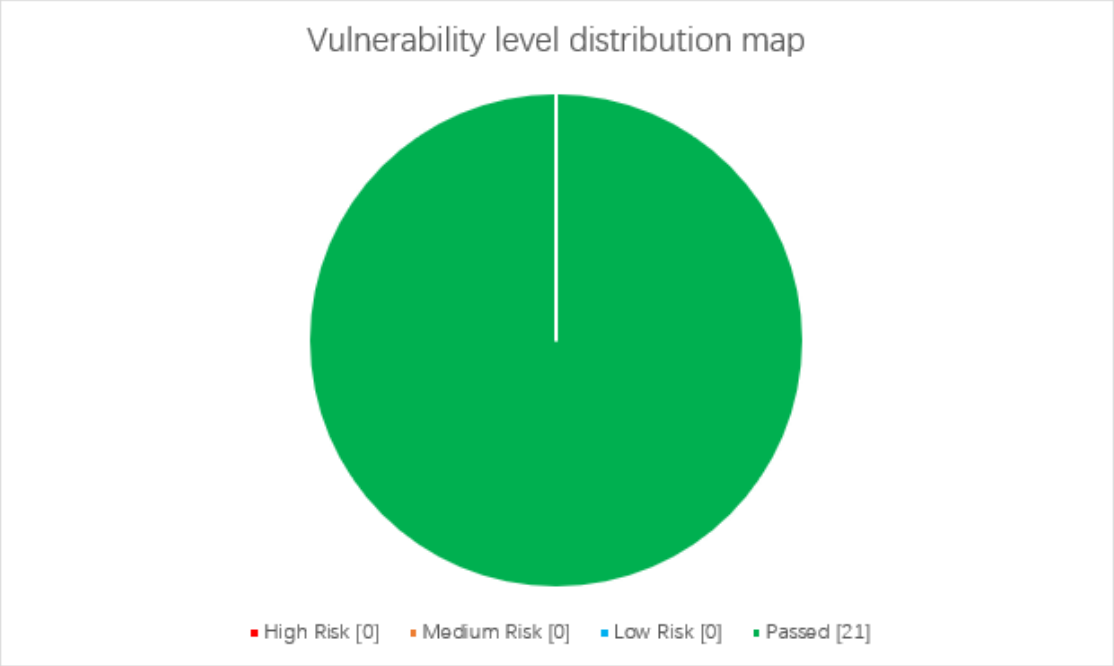
- deposit()
- withdraw(uint wad)
- totalSupply()
- approve(address guy, uint wad)
- mint(uint _value)
- transfer(address dst, uint wad)
- transferFrom(address src, address dst, uint wad)

4. Audit Details

4.1 Vulnerabilities Distribution

Vulnerabilities in this security audit are distributed by risk level, as follows:

Vulnerability level distribution			
High risk	Medium risk	Low risk	Passed
0	0	0	21



This smart contract security audit has 0 high-risk vulnerabilities, 0 medium-risk vulnerabilities, 0 low-risk vulnerabilities, and 21 passed, with a high security level.

4.2 Vulnerabilities Details

A security audit was conducted on the smart contract within the agreement, and no security vulnerabilities that could be directly exploited and generated security problems were found, and the security audit was passed.

4.3 Other Risks

Other risks refer to the code that contract security auditors consider to be risky, which may affect the stability of the project under certain circumstances, but cannot constitute a security issue that directly endangers the security.

4.3.1 Issue issue

- **Question detail**

In the WETH9 contract, there is a mint() function. After debugging, the contract has additional issuance problems. The code is as follows:

```
function mint(uint _value) public payable{
    balanceOf[msg.sender] += _value;
    Deposit(msg.sender, _value);
}
```

- **Safety advice**

It is recommended to delete sensitive functions in time.

- **Update status**

Through communication with the Powerswap team, the WETH9 contract is not used to issue coins, and the mint() function has no effect.

4.3.2 Safe function usage

- **Question detail**

In the WETH9 contract, the deposit() function, mint() function, and transferFrom() function all have arithmetic operations, and none of them use security functions, and there is a risk of integer overflow. The following code shows:

```
    function deposit() public payable {
        balanceOf[msg.sender] += msg.value;
        emit Deposit(msg.sender, msg.value);
    }
    function mint(uint _value)public payable{
        balanceOf[msg.sender] += _value;
        Deposit(msg.sender, _value);
    }
    function transferFrom(address src, address dst, uint wad) public returns
(bool){
        require(balanceOf[src] >= wad, "");
        if (src != msg.sender && allowance[src][msg.sender] != uint(-1)) {
            require(allowance[src][msg.sender] >= wad, "");
            allowance[src][msg.sender] -= wad;
        }
        balanceOf[src] -= wad;
        balanceOf[dst] += wad;
        emit Transfer(src, dst, wad);
        return true;
    }
```

- **Safety advice**

In order to eliminate the possibility of integer overflow, it is recommended to use the safe function safemath as much as possible in arithmetic operations.

- **Update status**

Through communication with the Powerswap team, the new function has been updated to include the safemath function.

4.3.3 Address validity check

- **Question detail**

In the WETH9 contract, the authorization function approve() and the transfer function transfer() both have address parameters passed in, and the validity of the passed address is not verified. If the passed address parameter is 0, it may cause the transfer of coins to be lost, etc. safe question.

```
    function approve(address guy, uint wad) public returns (bool) {
        allowance[msg.sender][guy] = wad;
        emit Approval(msg.sender, guy, wad);
        return true;
    }
    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

- **Safety advice**

Add address validity check, as shown in the following code:

```
require(Address != address(0) , 'ADDRESS ERROR!!!');
```

- **Update status**

Through communication with the Powerswap team, the address parameter has no effect.

4.3.4 Arithmetic problem one

- **Question detail**

In the PowerswapPair contract, the variable addition and subtraction operations in the _updateAfterSwap() function do not use safe functions, as shown in the following code:

```
    function _updateAfterSwap(
        uint amount0In,  uint amount1In,
        uint amount0Out, uint amount1Out,
        uint balance0,   uint balance1) private {
        if(amount0In > 0){
        balanceBuy0 += amount0In;
        balanceBuy1 -= amount1Out;
        }else{
        balanceSell0 -= amount0Out;
        balanceSell1 += amount1In;
        }
        reserve0 = uint112(balance0);
        reserve1 = uint112(balance1);
        emit Sync(reserve0, reserve1);
    }
```

- **Safety advice**

It is recommended to use SafeMath in the smart contract library to process arithmetic logic.

- **Update status**

Through communication with the Powerswap team, the addition operation is the addition of two uint112s, and there is no overflow; there will be an error when the subtraction operation overflows, and there is no security risk.

4.3.5 Arithmetic problem two

- **Question detail**

In the PowerswapPair contract, the variable addition and subtraction operation in the _updateAtTheFirstCallInEachBlock() function does not use a safe function, as shown in the following code:

```
function _updateAtTheFirstCallInEachBlock(uint112 _reserve0, uint112 _reserve1,
uint32 timeElapsed) private{
        balanceBuy0  = _reserve0;
        balanceBuy1  = _reserve1;
        balanceSell0 = _reserve0;
        balanceSell1 = _reserve1;
        uint _priceBuy  = priceBuy;
        uint _priceSell = priceSell;
        uint16 _R = R;
        priceBuy  = (_priceBuy.mul(_reserve1).mul(_R).add((uint(_reserve0) <<
RESOLUTION).mul(65535-_R)))/(_reserve1.mul(65535));
        priceSell = (_priceSell.mul(_reserve0).mul(_R).add((uint(_reserve1) <<
RESOLUTION).mul(65535-_R)))/(_reserve0.mul(65535));
        price0CumulativeLast +=
uint(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed;
        price1CumulativeLast +=
uint(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed;
    }
```

- **Safety advice**

It is recommended to use SafeMath in the smart contract library to process arithmetic logic.

- **Update status**

Through communication with the Powerswap team, the final calculation result did not carry out other operations and did not produce any impact.

4.3.6 Administrator authority is too large

- **Question detail**

In the PowerswapPair contract, the factory administrator can define the initial token address or set the R value. If the private key of the administrator address is stolen or manipulated by malicious people, the initialization function can be reused to define the token address or set the R value, resulting in Unconventional digital currency acquisition and market stability are shaken, as shown in the following code:

```
    function initialize(address _token0, address _token1) external override{
        require(msg.sender == factory, 'Powerswap: FORBIDDEN');
        token0 = _token0;
        token1 = _token1;
        R = 32767;  // default r = 0.5;
    }
    function setR(uint16 _R) external override {
        address _rSetter = IPowerswapFactory(factory).rSetter();
        require(msg.sender == _rSetter, 'Powerswap: FORBIDDEN');
        R = _R;
    }
```

- **Safety advice**

Keep multiple copies of the private key reasonably; you can set the initial token address to be called only once.

- **Update status**

After communicating with the Powerswap team, the team stated that it would properly manage the private key.

5. Security Audit Tool

Tool name	Tool Features
Oyente	Can be used to detect common bugs in smart contracts
securify	Common types of smart contracts that can be verified
MAIAN	Multiple smart contract vulnerabilities can be found and classified
Noneage Internal Toolkit	Noneage(hawkeye system) self-developed toolkit + https://audit.noneage.com

6. Vulnerability assessment criteria

Vulnerability level	Vulnerability description
High risk	Vulnerabilities that can directly lead to the loss of contracts or users' digital assets, such as integer overflow vulnerabilities, false recharge vulnerabilities, re-entry vulnerabilities, illegal token issuance, etc. Vulnerabilities that can directly cause the ownership change of the token contract or verification bypass, such as: permission verification bypass, call code injection, variable coverage, unverified return value, etc. Vulnerabilities that can directly cause the token to work normally, such as denial of service vulnerabilities, insecure random numbers, etc.
Medium risk	Vulnerabilities that require certain conditions to trigger, such as vulnerabilities triggered by the token owner's high authority, and transaction sequence dependent vulnerabilities. Vulnerabilities that cannot directly cause asset loss, such as function default visibility errors, logic design flaws, etc.
Low risk	Vulnerabilities that are difficult to trigger, or vulnerabilities that cannot lead to asset loss, such as vulnerabilities that need to be triggered at a cost higher than the benefit of the attack, cannot lead to incorrect coding of security vulnerabilities.

NONEAGE

Telephone：86-17391945345 18511993344

Email ：support@noneage.com

Site ：www.noneage.com

Weibo ：weibo.com/noneage