March 03, 2023

Share

How to Avoid Issues Related to Deflationary Tokens

Recent research by the Beosin security team found that security incidents caused by deflationary tokens are still frequent, resulting in the loss of funds for many projects. Therefore, we wrote this article on deflationary tokens to share with you.

1 What are Deflationary Tokens?

Deflationary tokens are tokens that are burned in proportion to the transaction process, which is a good way to incentivize users to hold tokens.

During the token trading process, some tokens will be deducted for fees, rewards and destruction, and as the tokens are burned, the total supply will keep decreasing, which will increase the proportion of tokens held by users, then users will be more willing to hold tokens to get higher returns.

It seems to be a perfect financial solution, but there is a burning process in the code implementation, which will bypass the swap process and modify the address balance directly. Some unexpected issues may occur when this situation is combined with pair.

2 What are the Issues Exist in Deflationary Tokens?

(1) Add liquidity

The deflationary token will charge a certain percentage of fee to the current contract when transferring, and the pair contract will be called to swap, addLiquidity or sync when the fee reaches a certain threshold (the current number of tokens is greater than or equal to a variable set by the contract).

Related Project

Related Project S
Score

Learn More

Guess you like


Beosin Become
ive Security Au
March 01, 2023


Beosin and Has
e Strengthened
March 03, 2023


Summary of SU
minar 'The Evol
March 02, 2023


Blockchain Secu
hly Recap of Fe
March 02, 2023

Join the commun
to discuss.

Why does the transaction fail? Adding liquidity is done by depositing both TokenA and TokenB tokens into the pair contract, and then calling the mint function of the pair contract, which determines how many tokens the user has passed in based on the difference between the current balance and the reserve of this contract.

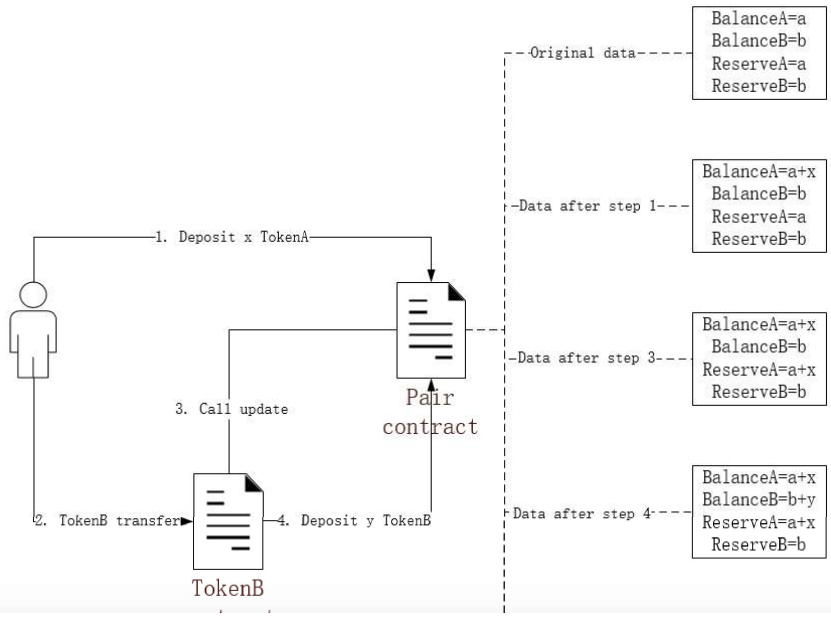
After the user sends TokenA tokens to pair and makes a TokenB transfer, if the fee charged happens to reach the above threshold, the token contract calls the swap, mint or sync function of pair, which will all call the _update function of pair, thus updating the TokenA that the user initially sent to pair to reserve.

Finally, the user calls the mint function, which will cause TokenA's balance and reserve to be equal, and as a result will cause the transaction to fail.

Mint function:

```
function mint(address to) external lock returns (uint liquidity)
{
(uint112 _reserve0, uint112 _reserve1,) = getReserves(); //
uint balance0 = IERC20(token0).balanceOf(address(this));// Get TokenA balance
uint balance1 = IERC20(token1).balanceOf(address(this));// Get TokenB balance
uint amount0 = balance0.sub(_reserve0);// The difference is 0
uint amount1 = balance1.sub(_reserve1);
bool feeOn = _mintFee(_reserve0, _reserve1);
uint _totalSupply = totalSupply;
if (_totalSupply == 0) {
liquidity = Math.sqrt(amount0.mul(amount1)).sub(MINIMUM_LIQUIDITY);
_mint(address(0), MINIMUM_LIQUIDITY);
} else {
liquidity = Math.min(amount0.mul(_totalSupply) / _reserve0, amount1.mul(_totalSupply) /
_reserve1);
}
require(liquidity > 0, 'UniswapV2: INSUFFICIENT_LIQUIDITY_MINTED');// Here will cause a
transaction failure
_mint(to, liquidity);
_update(balance0, balance1, _reserve0, _reserve1);
if (feeOn)
kLast = uint(reserve0).mul(reserve1);
emit Mint(msg.sender, amount0, amount1);
}
```

The call process:



(2) Skim function

The Pair contract has a skim function that sends tokens in the pair contract that are in excess of the reserve to the address specified by the caller, and the number is calculated based on the difference between the number of tokens in the pair contract and the reserve. This in itself is a function of balancing the pair supply, but problems can arise when one of the tokens is a deflationary token.

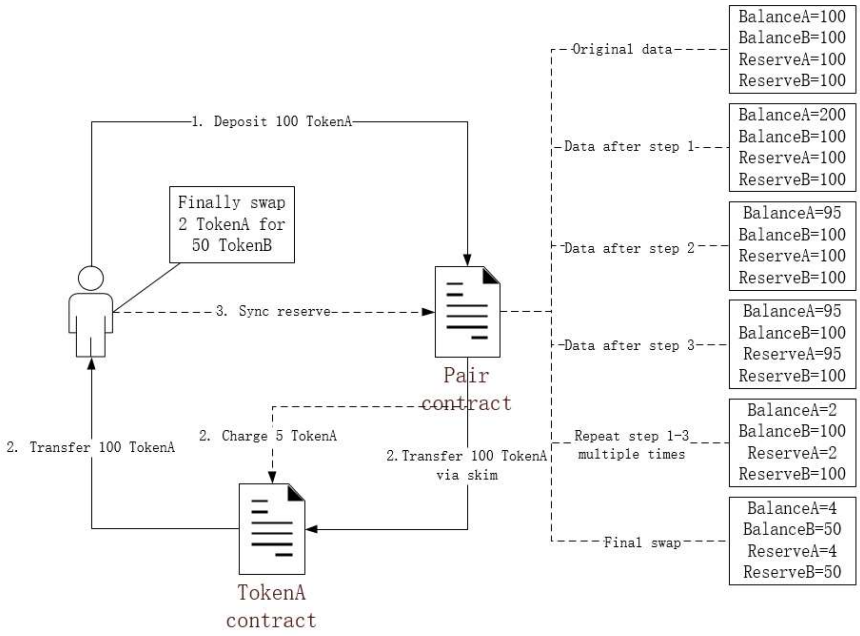
Deflationary tokens will deduct a certain fee in the transaction, so what happens if the fees deducted during the token transfer are paid by _from address in the skim function?

At this point, the fee deducted will be the supply of the pair, so that the tokens can be transferred to the pair in advance, through the continuous skim function and sync function to consume the supply of the pair, the price of the tokens in the pair continues to soar, and eventually a small number of the deflationary tokens can be exchanged for a large number of another token (usually usdt, eth).

Skim function:

```
function skim(address to) external lock {
address _token0 = token0;
  address _token1 = token1;
  _safeTransfer(_token0,to,IERC20(_token0).balanceOf(address(this)).sub(reserve0));
  _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}
```

The whole process:



(3) Burn

This problem is mainly found in deflationary tokens that use a "reflection" mechanism,

What is the role of rOwned? As mentioned at the beginning of the article, deflationary tokens provide an incentive for users to hold tokens, and the way this incentive is used is to deduct the rOwned value from the trader, and at the same time deduct rTotal, so that the ratio of rOwned to rTotal for other users will be passively increased to achieve passive gains. (rOwned and rTotal can be interpreted as the user's shares and total shares)

The user can query the balance in two ways, one is the excluded address, which directly returns the value of tOwned, and the other is the non-excluded address, which returns rOwned/currentRate, and currentRate is calculated as rTotal/tTotal. If there is a way to make rTotal decrease, the actual balance queried by the user will become larger. If the pair query balance becomes larger, then the excess tokens can be transferred out through the skim function.

There exists a deliver() function for this class of deflationary tokens, which can be called by non-excluded addresses. The function will burn the rOwned of the caller and burn the same number of _rTotal, making the balance query increase for all non-excluded addresses. If pair is non-excluded, then the arbitrage attack can be used in the above manner.

3 Related Security Incidents

(1) AES
https://twitter.com/BeosinAlert/status/1600478366255427584

(2) Crypto1319, The Philosophers Stone ($TPOS) and Meta Speed Game ($MTSG)
https://twitter.com/BeosinAlert/status/1623942545540788225

(3) TATA
https://twitter.com/BeosinAlert/status/1628636827279331330

(4) FDP
https://twitter.com/BeosinAlert/status/1622806011269771266

Contact

If you have need any blockchain security services, please contact us: