Coinsult

Advanced Manual Smart Contract Audit

Project: Criba Inu (Router)

Website: https://www.cribainu.com/

Low-Risk	Medium-Risk	High-Risk
5 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x31d21Ec3722Cf628380b330ca01fc71A5F5e4c65

Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

-

Source Code

Coinsult was comissioned by Criba Inu (Router) to perform an audit based on the following smart contract:

https://cronoscan.com/address/0x31d21ec3722cf628380b330ca01fc71a5f5e4c65#code

Contract is setup as a router

Manual Code Review

In this audit report we will highlight all these issues:

Low-Risk

5 low-risk code
issues found

Medium-Risk

0 medium-risk code
issues found

High-Risk

0 high-risk code
issues found

The detailed report continues on the next page...

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
modifier ensure(uint deadline) {
    require(deadline &gt;= block.timestamp, 'RouterV2: EXPIRED');
    _;
}
```

Recommendation

Do not use `block.timestamp`, now or `blockhash` as a source of randomness

Exploit scenario

```
contract Game {

    uint reward_determining_number;

    function guessing() external{
      reward_determining_number = uint256(block.blockhash(10000)) % 10;
    }
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
{
    require(path[0] == WETH, 'RouterV2: INVALID_PATH');
    uint amountIn = msg.value;
    IWETH(WETH).deposit{value: amountIn}();
    assert(IWETH(WETH).transfer(MainLibrary.pairFor(factory, path[0], path[1]), amountIn));
    uint balanceBefore = IERC20(path[path.length - 1]).balanceOf(to);
    _swapSupportingFeeOnTransferTokens(path, to);
    require(
        IERC20(path[path.length - 1]).balanceOf(to).sub(balanceBefore) &gt;= amountOutMin,
        'RouterV2: INSUFFICIENT_OUTPUT_AMOUNT'
    );
}
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

`updateOwner()` has no event, so it is difficult to track off-chain changes in the buy price.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Conformance to Solidity naming conventions

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
IWETH(WETH).withdraw(amountETH);
```

Recommendation

Follow the Solidity naming convention.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow _ at the beginning of the `mixed_case` match for private variables and unused parameters.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Redundant Statements

Detect the usage of redundant statements that have no effect.

```
// refund dust eth, if any
if (msg.value &gt; amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);

(remove comments)
```

Recommendation

Remove redundant statements if they congest code but offer no value.

Exploit scenario

```
contract RedundantStatementsContract {

    constructor() public {
        uint; // Elementary Type Name
        bool; // Elementary Type Name
        RedundantStatementsContract; // Identifier
    }

    function test() public returns (uint) {
        uint; // Elementary Type Name
        assert; // Identifier
        test; // Identifier
        return 777;
    }
}
```

Each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements and they can be removed.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Costly operations inside a loop

Costly operations inside a loop might waste gas, so optimizations are justified.

```
function _swap(uint[] memory amounts, address[] memory path, address _to) internal virtual {
    for (uint i; i &lt; path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0,) = MainLibrary.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, ui
        address to = i &lt; path.length - 2 ? MainLibrary.pairFor(factory, output, path[i + 2]) : _t
        IUniswapV2Pair(MainLibrary.pairFor(factory, input, output)).swap(
            amount0Out, amount1Out, to, new bytes(0)
        );
    }
}
```

Recommendation

Use a local variable to hold the loop computation result.

Exploit scenario

```
contract CostlyOperationsInLoop{

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTORE`s, which might lead to an `out-of-gas`.

Owner privileges

- No real owner privileges

Extra notes by the team

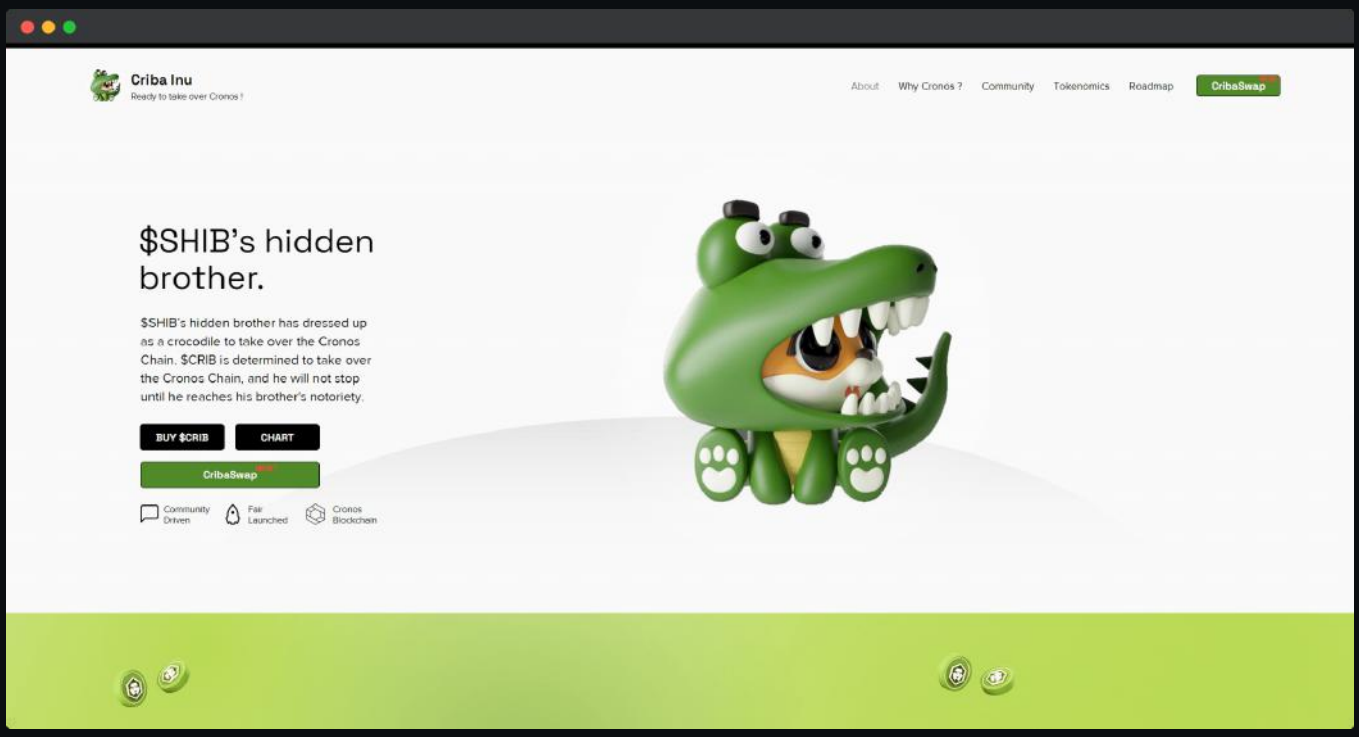
No notes

Contract Snapshot

```
contract RouterV2 is IUniswapV2Router02 {
using SafeMath for uint;

address public immutable override factory;
address public immutable override WETH;

modifier ensure(uint deadline) {
    require(deadline &gt;= block.timestamp, 'RouterV2: EXPIRED');
    _;
}

constructor(address _factory, address _WETH) {
    factory = _factory;
    WETH = _WETH;
}

receive() external payable {
    assert(msg.sender == WETH); // only accept ETH via fallback from the WETH contract
}

// **** ADD LIQUIDITY ****
function _addLiquidity(
    address tokenA,
    address tokenB,
    uint amountADesired,
    uint amountBDesired,
    uint amountAMin,
    uint amountBMin
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



- Mobile Friendly
- Does not contain jQuery errors
- SSL Secured
- No major spelling errors

Project Overview

Not KYC verified by Coinsult