Coinsult

Advanced Manual Smart Contract Audit

Project: Larva Token

Website: https://larvainu.io/

● 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

0xb9E6Bc08Db15c7164b56e7c6642210c8D1A5c302

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

Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	553,650,000	55.3650%
2	0xf33197dfe08ae842e19cde79880db77f8848e6b3	445,350,000	44.5350%
3	0x407993575c91ce7643a4d4ccacc9a98c36ee1bbe	1,000,000	0.1000%

Source Code

Coinsult was comissioned by Larva Token to perform an audit based on the following smart contract:

https://bscscan.com/address/0xb9E6Bc08Db15c7164b56e7c6642210c8D1A5c302#code

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.

Contract contains Reentrancy vulnerabilities

Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback).

More information: Slither

```
function _transfer(
    address from,
    address to,
    uint256 amount
) private {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(amount &gt; 0, "Transfer amount must be greater than zero");

     if(from != owner() &amp;&amp; to != owner() &amp;&amp; to != uniswapV2Pair)
        require(balanceOf(to) + amount &lt;= _maxTxAmount, &quot;Transfer amount exceeds the maxTxAm


    if (from == uniswapV2Pair) {
        require (_lastBuy[to] + _buyCooldown = SWAP_TOKENS_AT_AMOUNT &amp;&amp; !swapping &amp;&amp;
        swapping = true;
        uint256 sellTokens = balanceOf(address(this));
        swapAndSendToFee(sellTokens);
        swapping = false;
    }

     tOwned[from] -= amount;
```

Recommendation

Apply the check-effects-interactions pattern.

Exploit scenario

```
function withdrawBalance(){
    // send userBalance[msg.sender] Ether to msg.sender
    // if mgs.sender is a contract, it will call its fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
        throw;
    }
    userBalance[msg.sender] = 0;
}
```

Bob uses the re-entrancy bug to call withdrawBalance two times, and withdraw more than its initial deposit to the contract.

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

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
if (from == uniswapV2Pair) {
    require (_lastBuy[to] + _buyCooldown &lt; block.timestamp, &quot;Must wait til after coooldown t
    _lastBuy[to] = block.timestamp;
}
```

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.

Too many digits

Literals with many digits are difficult to read and review.

```
uint256 private _tTotal = 1000000000 * 10**_decimals;
uint256 public  _maxTxAmount = 1000000000 * 10**_decimals;          // balance of receiver after bu
uint256 private constant SWAP_TOKENS_AT_AMOUNT = 3 * 10**_decimals; // minimum sto swap token to bnb
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

While `1_ether` looks like `1 ether`, it is `10 ether`. As a result, it's likely to be used incorrectly.

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function update_Tax(uint256 taxBuy, uint256 taxSell, uint256 taxSell_marketing, uint256 taxSell_dev,
    require(taxBuy&lt;25 &amp;&amp;  taxSell&lt;25 &amp;&amp; taxSell_marketing&lt;100 &amp;&amp; ta
    _taxBuy = taxBuy;
    _taxSell = taxSell;
    _slotMarketing = taxSell_marketing;
    _slotDev = taxSell_dev;
    _slotFund = taxSell_fund;
    _liquidityFee = liquidityFee;
}
```

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.

```
function update_buyCooldown(uint256 newWaitingTime) public onlyOwner{
    _buyCooldown = newWaitingTime;
}
```

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.

Owner privileges

- 🟢 Owner cannot change max transaction amount
- 🟡 Owner can set fees higher than 25%
- 🟡 Owner can exclude from fees
- 🟡 Owner can pause the contract

Extra notes by the team

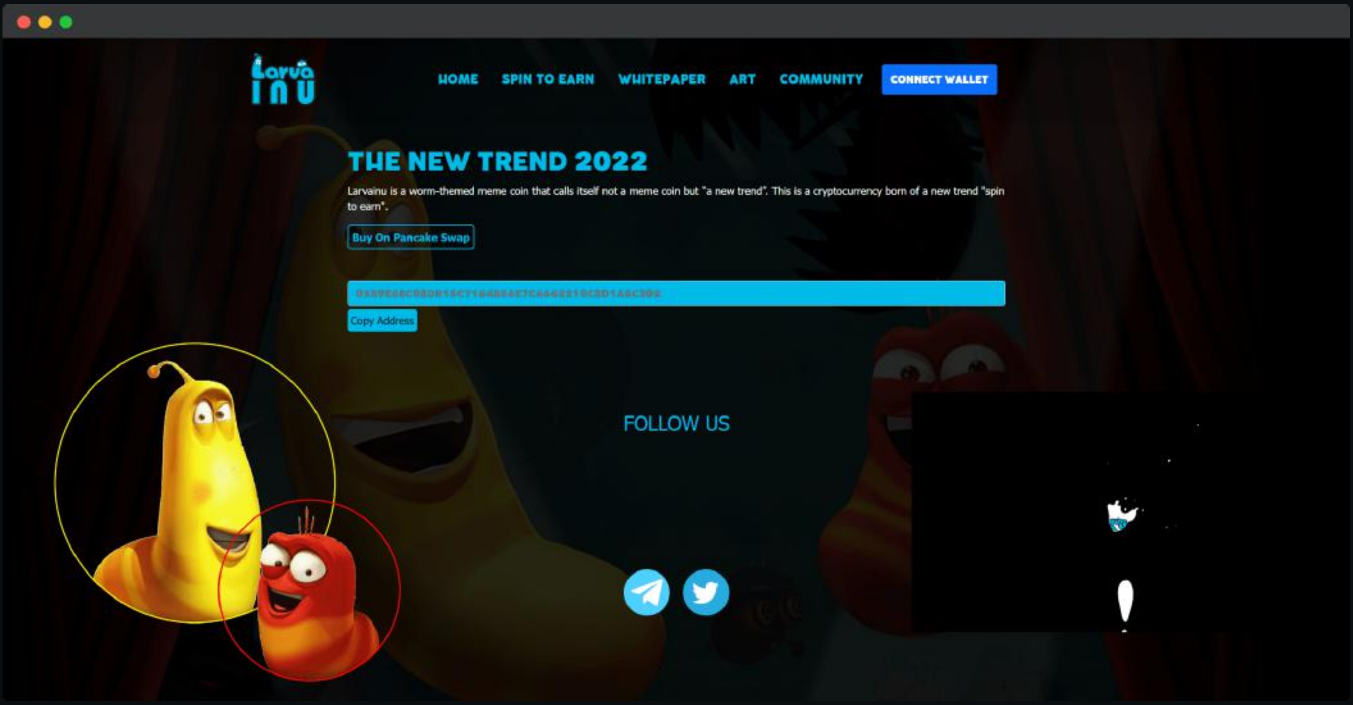
No notes

Contract Snapshot

```
contract LAV is Context, IERC20, Ownable {

string private constant _name = "LARVA TOKEN";
string private constant _symbol = "LAV";
uint8 private constant _decimals = 18;

uint256 private _tTotal = 1000000000 * 10**_decimals;
uint256 public  _maxTxAmount = 1000000000 * 10**_decimals;           // balance of receiver after bu
uint256 private constant SWAP_TOKENS_AT_AMOUNT = 3 * 10**_decimals; // minimum sto swap token to bnb
uint256 private _liquidityFee;

uint256 private  _taxBuy             = 8;          // tax apply on user buying token
uint256 private  _taxSell            = 8;           // tax apply on user buying token
uint256 private  _slotMarketing      = 20;          // amount of token swap to bnb for marketing
uint256 private  _slotDev            = 70;          // amount of token swap to bnb for developer
uint256 private  _slotFund           = 10;          // amount of token swap to bnb for fund
address private  _marketingWallet    = 0xfFdd6fc6560F66A8A6ff377614BB72f03cEBac7d;
address private  _devWallet          = 0x60475b418B97e2D0E7442472bC9EEbE96e9A518C ;
address private  _fundWallet         = 0xbc8E5037af3c3FdA72F481cFB63ED42A5fCEbE4b;

IUniswapV2Router02 public uniswapV2Router;
address public uniswapV2Pair;
using Address for address;
using Address for address payable;
bool private swapping;
mapping (address =&gt; uint256) private _tOwned;
mapping (address =&gt; mapping (address =&gt; uint256)) private _allowances;
mapping (address =&gt; bool) private _isExcludedFromFee;
uint256 public _buyCooldown = 0 minutes;
mapping (address =&gt; uint256) private _lastBuy;

event SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);

constructor () {
    _tOwned[_msgSender()] = _tTotal;

    // 0x10ED43C718714eb63d5aA57B78B54704E256024E //PancakeSwap v2 router
```

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.



THE NEW TREND 2022

Larvainu is a worm-themed meme coin that calls itself not a meme coin but "a new trend". This is a cryptocurrency born of a new trend "spin to earn".

[illegible]

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

Project Overview

● Not KYC verified by Coinsult

Larva Token

Audited by Coinsult.net

Date: 22 June 2022

✓ Advanced Manual Smart Contract Audit