Coinsult

Advanced Manual Smart Contract Audit

Project: Amano
Website: https://amano.financial

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

Contract address
Not deployed yet

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 commissioned by AmanoFinancial to perform an audit based on the following smart contract:

Private source code

Manual Code Review

🟢 Low-risk

4 low-risk code issues found.
Could be fixed, will not bring problems.

- Contract contains Reentrancy vulnerabilities: _transferFrom(address,address,uint256)

  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 _transferFrom(address sender, address recipient, uint256 amount) internal returns (bool) {
        bool excludedAccount = _isFeeExempt[sender] || _isFeeExempt[recipient];

        require(initialDistributionFinished || excludedAccount, "Trading not started");

        if (
            automatedMarketMakerPairs[recipient] &&
            !excludedAccount
        ) {
            require(amount <= maxSellTransactionAmount, "Error amount");
        }

        if (inSwap) {
            return _basicTransfer(sender, recipient, amount);
        }

        uint256 gonAmount = amount.mul(_gonsPerFragment);

        if (shouldSwapBack() && recipient!= DEAD) {
            swapBack();
        }

        _gonBalances[sender] = _gonBalances[sender].sub(gonAmount);

        uint256 gonAmountReceived = shouldTakeFee(sender, recipient) ? takeFee(sender, recipient, gonAmount) :
gonAmount;

        _gonBalances[recipient] = _gonBalances[recipient].add(gonAmountReceived);

        emit Transfer(
            sender,
            recipient,
            gonAmountReceived.div(_gonsPerFragment)
        );

        if(shouldRebase() && recipient!= DEAD) {
            _rebase();
        }

        return true;
    }
```

- Block.timestamp can be manipulated by miners.
  Avoid relying on block.timestamp.

  More information:
  https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

```
    uint256 public nextRebase = block.timestamp + 101536000;
```

- Literals with many digits are difficult to read and review.
  Recommendation: Use Ether suffix, Time suffix, or The scientific notation

```
    uint256 public rewardYield = 2291667;
    uint256 public rewardYieldDenominator = 10000000000;
    uint256 public maxSellTransactionAmount = 250000 * 10 ** 18;
```

- Missing zero address validation
  Check that the new address is not zero.

```
    function setFeeReceivers(address _liquidityReceiver, address
_treasuryReceiver, address _amanoBuybackAssuranceFundReciver) external
onlyOwner {
        liquidityReceiver = _liquidityReceiver;
        treasuryReceiver = _treasuryReceiver;
        amanoBuybackAssuranceFundReciver =
_amanoBuybackAssuranceFundReciver;
    }
```

🟡 Medium-risk

0 medium-risk code issues found.
Should be fixed, could bring problems.

🔴 High-risk

0 high-risk code issues found
Must be fixed, and will bring problems.

Extra notes by the team

🟡 Owner can set swap to disabled

Note from the team: Like for example exploitation happens or any alike circumstances. We can adhere and stop it right away. And protect our investors' investments safe and sound.

```
    function setSwapBackSettings(bool _enabled, uint256 _num, uint256 _denom)
external onlyOwner {
        swapEnabled = _enabled;
        gonSwapThreshold = TOTAL_GONS.div(_denom).mul(_num);
    }
```

🟡 Owner can set max sell transaction to anything higher than 0

Note from the team: We have to limit the whales from dumping and hence we have this feature to change the limit of the sell amount

```
    function setMaxSellTransaction(uint256 _maxTxn) external onlyOwner {
        require(_maxTxn != 0,"cannot be 0");
        maxSellTransactionAmount = _maxTxn;
    }
```

🟡 Total buy fees can not be greater than 25%, which is passed by as a variable. `totalSellFee` can be greater than 25% not greater than 45%.

Note from the team: To stop selling we have kept the feature to keep high selling fees if needed.

```
    function setFees(uint256 _liquidityFee, uint256 _amanoAssuranceValue, uint256 _treasuryFee,
uint256 _sellFeeTreasuryAdded, uint256 _sellFeeABAAdded) external onlyOwner {
        require(
            _liquidityFee <= MAX_FEE_RATE &&
            _amanoAssuranceValue <= MAX_FEE_RATE &&
            _treasuryFee <= MAX_FEE_RATE &&
            _sellFeeTreasuryAdded <= MAX_FEE_RATE &&
            _sellFeeABAAdded <= MAX_FEE_RATE,
            "wrong"
        );

        liquidityFee = _liquidityFee;
        buyFeeABA = _amanoAssuranceValue;
        treasuryFee = _treasuryFee;
        sellFeeTreasuryAdded = _sellFeeTreasuryAdded;
        sellFeeABAAdded = _sellFeeABAAdded;
        totalBuyFee = liquidityFee.add(treasuryFee).add(buyFeeABA).add(burnFeeBuy);
        totalSellFee = totalBuyFee.add(sellFeeTreasuryAdded).add(sellFeeABAAdded).add(burnFeeSell);
        require(totalBuyFee <= feeDenominator / 4);
        require(totalSellFee <= 45);
    }
```

Contract Snapshot

```
abstract contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor(
        string memory _tokenName,
        string memory _tokenSymbol,
        uint8 _tokenDecimals
    ) {
        _name = _tokenName;
        _symbol = _tokenSymbol;
        _decimals = _tokenDecimals;
    }

    function name() public view returns (string memory) {
        return _name;
    }

    function symbol() public view returns (string memory) {
        return _symbol;
    }

    function decimals() public view returns (uint8) {
        return _decimals;
    }
}
```

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
- Contains no jQuery errors
- SSL Secured
- No major spelling errors

Loading speed: 92%

Rug-pull Review

Based on the available information analyzed by us, we come to the following conclusions:

- 🟢 Locked Liquidity - No liquidity yet
- 🟢 Large unlocked wallets - Tokens not yet distributed
- 🟢 Doxxed Team (KYC at Coinsult)

Honeypot Review

Based on the available information analyzed by us, we come to the following conclusions:

- 🟡 Ability to sell
  `totalSellFee` can be greater than 25% but not greater than 45%.
- 🟡 Owner is able to pause trading
- 🟢 Router hard coded in the contract

Note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.