Coinsult

Advanced Manual Smart Contract Audit

Project: XenoVerse
Website: https://xenoversecrypto.com

● Low-risk
6 low-risk code issues found

● Medium-risk
0 medium-risk code issues found

● High-risk
0 high-risk code issues found

Contract address
0x8B6C453508F96B1946c2CFB28E2648c498abB414

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	0xb64dba2200a5b863f9aa0efb4b5fc66cb6d51172	80,784,000	80.7840%
2	PinkSale: PinkLock	14,216,000	14.2160%
3	0x9934c1f46702dfe29b5925cf9cbb73fd4f062e80	5,000,000	5.0000%

Source code

Coinsult was commissioned by XenoVerse to perform an audit based on the following smart contract:

https://bscscan.com/address/0x8B6C453508F96B1946c2CFB28E2648c498abB414#code

Manual Code Review

● Low-risk

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

- Usage of block.timestamp. block.timestamp can be manipulated by miners

  Recommendation: Avoid relying on block.timestamp.

```
    function canAutoClaim(uint256 lastClaimTime) private view returns
(bool) {
        if (lastClaimTime > block.timestamp) {
            return false;
        }

        return block.timestamp.sub(lastClaimTime) >= claimWait;
    }
```

- 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
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero
address");
        require(to != address(0), "ERC20: transfer to the zero
address");

        if (enableAntiBot) {
            pinkAntiBot.onPreTransferCheck(from, to, amount);
        }
        if (amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if (
            canSwap &&
            !swapping &&
            !automatedMarketMakerPairs[from] &&
            from != owner() &&
            to != owner()
        ) {
            swapping = true;

            uint256 marketingTokens = contractTokenBalance
                .mul(marketingFee)
                .div(totalFees);
            swapAndSendToFee(marketingTokens);

            uint256 swapTokens =
contractTokenBalance.mul(liquidityFee).div(
                totalFees
            );
            swapAndLiquify(swapTokens);

            uint256 sellTokens = balanceOf(address(this));
            swapAndSendDividends(sellTokens);

            swapping = false;
        }

        bool takeFee = !swapping;

        // if any account belongs to _isExcludedFromFee account then
remove the fee
        if (_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
            takeFee = false;
        }
        if (takeFee) {
            uint256 fees = amount.mul(totalFees).div(100);
            if (automatedMarketMakerPairs[to]) {
                fees += amount.mul(1).div(100);
            }
            amount = amount.sub(fees);

            super._transfer(from, address(this), fees);
        }

        super._transfer(from, to, amount);

        try
            dividendTracker.setBalance(payable(from), balanceOf(from))
        {} catch {}
        try dividendTracker.setBalance(payable(to), balanceOf(to)) {}
catch {}

        if (!swapping) {
            uint256 gas = gasForProcessing;

            try dividendTracker.process(gas) returns (
                uint256 iterations,
                uint256 claims,
                uint256 lastProcessedIndex
            ) {
                emit ProcessedDividendTracker(
                    iterations,
                    claims,
                    lastProcessedIndex,
                    true,
                    gas,
                    tx.origin
                );
            } catch {}
        }
    }
```

- To many digits (Use: Ether suffix, Time suffix, or The scientific notation)

```
        require(
            newValue >= 200000 && newValue <= 500000,
            "BABYTOKEN: gasForProcessing must be between 200,000 and
500,000"
        );
```

- The return value of an external transfer/transferFrom call is not checked

  Recommendation: Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

```
 function swapAndSendToFee(uint256 tokens) private {
        uint256 initialCAKEBalance = IERC20(rewardToken).balanceOf(
            address(this)
        );

        swapTokensForCake(tokens);
        uint256 newBalance =
(IERC20(rewardToken).balanceOf(address(this))).sub(
            initialCAKEBalance
        );
        IERC20(rewardToken).transfer(_marketingWalletAddress,
newBalance);
    }
```

- No zero address validation

  Check that the new address is not the zero address

```
    function updateUniswapV2Router(address newAddress) public onlyOwner
{
        require(
            newAddress != address(uniswapV2Router),
            "BABYTOKEN: The router already has that address"
        );
        emit UpdateUniswapV2Router(newAddress,
address(uniswapV2Router));
        uniswapV2Router = IUniswapV2Router02(newAddress);
        address _uniswapV2Pair =
IUniswapV2Factory(uniswapV2Router.factory())
            .createPair(address(this), uniswapV2Router.WETH());
        uniswapV2Pair = _uniswapV2Pair;
    }
```

- Contract is named BabyToken in code

```
contract BABYTOKENDividendTracker is OwnableUpgradeable,
DividendPayingToken {
```

🟡 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

🟢 Contract has anti-bot system

🟢 Owner can not set the fees higher than 25%

```
    function setTokenRewardsFee(uint256 value) external onlyOwner {
        tokenRewardsFee = value;
        totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }

    function setLiquiditFee(uint256 value) external onlyOwner {
        liquidityFee = value;
        totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }

    function setMarketingFee(uint256 value) external onlyOwner {
        marketingFee = value;
        totalFees =
tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }
```

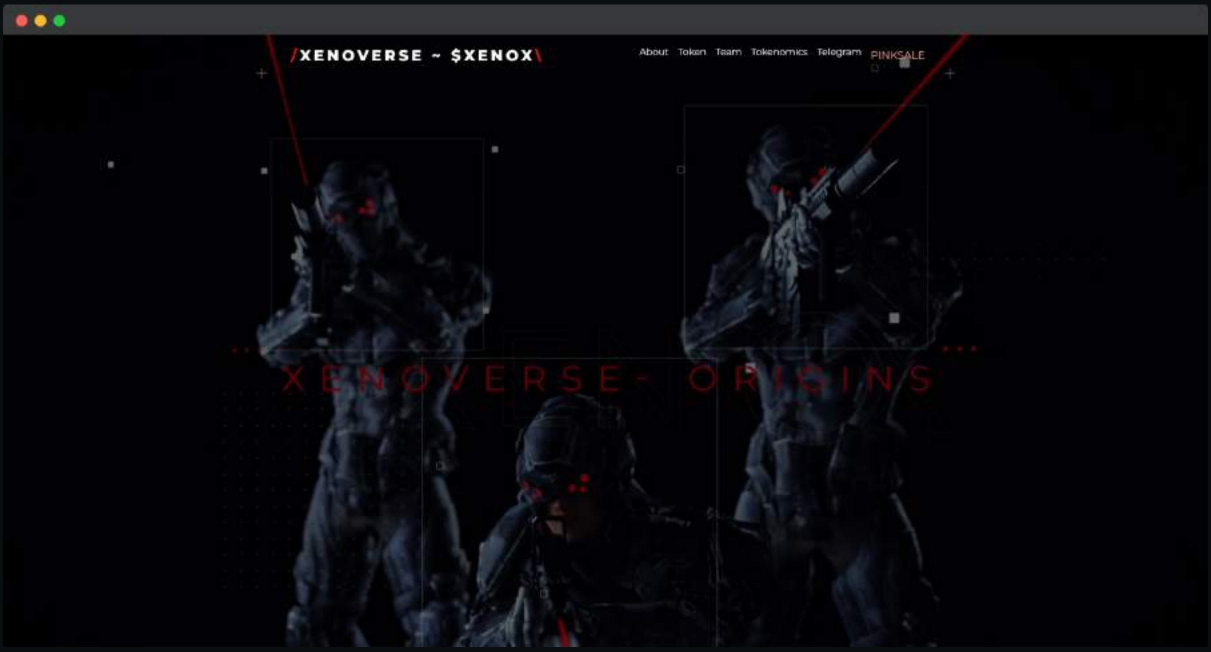
🟡 Owner can exclude from fees

🟡 Ownership is not renounced

Contract Snapshot

```
contract BABYTOKENDividendTracker is OwnableUpgradeable,
DividendPayingToken {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using IterableMapping for IterableMapping.Map;

    IterableMapping.Map private tokenHoldersMap;
    uint256 public lastProcessedIndex;

    mapping(address => bool) public excludedFromDividends;

    mapping(address => uint256) public lastClaimTimes;

    uint256 public claimWait;
    uint256 public minimumTokenBalanceForDividends;

    event ExcludeFromDividends(address indexed account);
    event ClaimWaitUpdated(uint256 indexed newValue, uint256 indexed
oldValue);

    event Claim(
        address indexed account,
        uint256 amount,
        bool indexed automatic
    );

    function initialize(
        address rewardToken_,
        uint256 minimumTokenBalanceForDividends_
    ) external initializer {
        DividendPayingToken.__DividendPayingToken_init(
            rewardToken_,
            "DIVIDEND_TRACKER",
            "DIVIDEND_TRACKER"
        );
        claimWait = 3600;
        minimumTokenBalanceForDividends =
minimumTokenBalanceForDividends_;
    }
```

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: 90%

Rug-pull Review

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

🟢 Locked Liquidity

🟢 Large unlocked wallets

- Note: Tokens not distributed yet

🟡 No doxxed Team

Honeypot Review

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

🟢 Ability to sell

🟢 Owner is not able to pause the contract

🟡 Router can be changed

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.