Coinsult

Advanced Manual Smart Contract Audit

Project: Birdman Token

Website: https://www.birdman.fashion

🟢 **Low-Risk**	🟡 **Medium-Risk**	🔴 **High-Risk**
4 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x094956520ef90526F888b5057A830089742D7ff0

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	890,000,000,000,000	89.0000%
2	0x636df130eef84549c0ece55dd60360a01ac3d6c1	59,598,957,000,000	5.9599%
3	0x95f296390d8cdca9fbf255c2af1f29ce3fdb1676	45,401,043,000,000	4.5401%
4	0x6c57c8cb904aaac8add12e0d86c7680512611476	5,000,000,000,000	0.5000%

Source Code

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

https://bscscan.com/address/0x094956520ef90526F888b5057A830089742D7ff0#code

Modified PinkSale BABYTOKEN contract

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

4 low-risk code
issues found

🟡 Medium-Risk

1 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

```
External calls sending eth:
- swapAndLiquify(AmountLiquidityFee) (#2229)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.t
State variables written after the call(s):
- AmountLiquidityFee += LFee (#2250)
- AmountLiquidityFee += LFee (#2260)
- AmountMarketingFee += MFee (#2254)
- AmountMarketingFee += MFee (#2264)
- AmountTokenRewardsFee += RFee (#2252)
- AmountTokenRewardsFee += RFee (#2262)
- super._transfer(from,deadWallet,DFee) (#2269)
- _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (#569)
- _balances[recipient] = _balances[recipient].add(amount) (#570)
- super._transfer(from,address(this),fees.sub(DFee)) (#2270)
- _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (#569)
- _balances[recipient] = _balances[recipient].add(amount) (#570)
- super._transfer(from,to,amount) (#2273)
- _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (#569)
- _balances[recipient] = _balances[recipient].add(amount) (#570)
- swapping = false (#2231)
```

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.

No zero address validation for some functions

Detect missing zero address validation.

```
function setMarketingWallet(address payable wallet) external onlyOwner{
    _marketingWalletAddress = wallet;
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

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

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

Bob calls `updateOwner` without specifying the `newOwner`, soBob loses ownership of the contract.

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

Unchecked transfer

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

```
function swapManual() public onlyOwner {
    uint256 contractTokenBalance = balanceOf(address(this));
    require(contractTokenBalance &gt; 0 , "token balance zero");
    swapping = true;
    if(AmountLiquidityFee &gt; 0) swapAndLiquify(AmountLiquidityFee);
    if(AmountTokenRewardsFee &gt; 0) swapAndSendDividends(AmountTokenRewardsFee);
    if(AmountMarketingFee &gt; 0) swapAndSendToFee(AmountMarketingFee);
    swapping = false;
}
```

Recommendation

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

Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setBuyTaxes(uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee) e
    require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) &lt;= 25, &quot;Total buy fee i
    buyTokenRewardsFee = rewardsFee;
    buyLiquidityFee = liquidity;
    buyMarketingFee = marketingFee;
    buyDeadFee = deadFee;

}

function setSelTaxes(uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee) e
    require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) &lt;= 25, &quot;Total sel fee i
    sellTokenRewardsFee = rewardsFee;
    sellLiquidityFee = liquidity;
    sellMarketingFee = marketingFee;
    sellDeadFee = deadFee;
}
```

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.

● **Medium-Risk:** Should be fixed, could bring problems.

No reason to have a ‘setDeadWallet’ function

```
function setDeadWallet(address addr) public onlyOwner {
    deadWallet = addr;
}
```

Recommendation

The dead wallet is static and does not ever have to be changed. For transparancy, remove this function to be sure all burn fees go to the dead wallet.

Owner privileges

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

Extra notes by the team

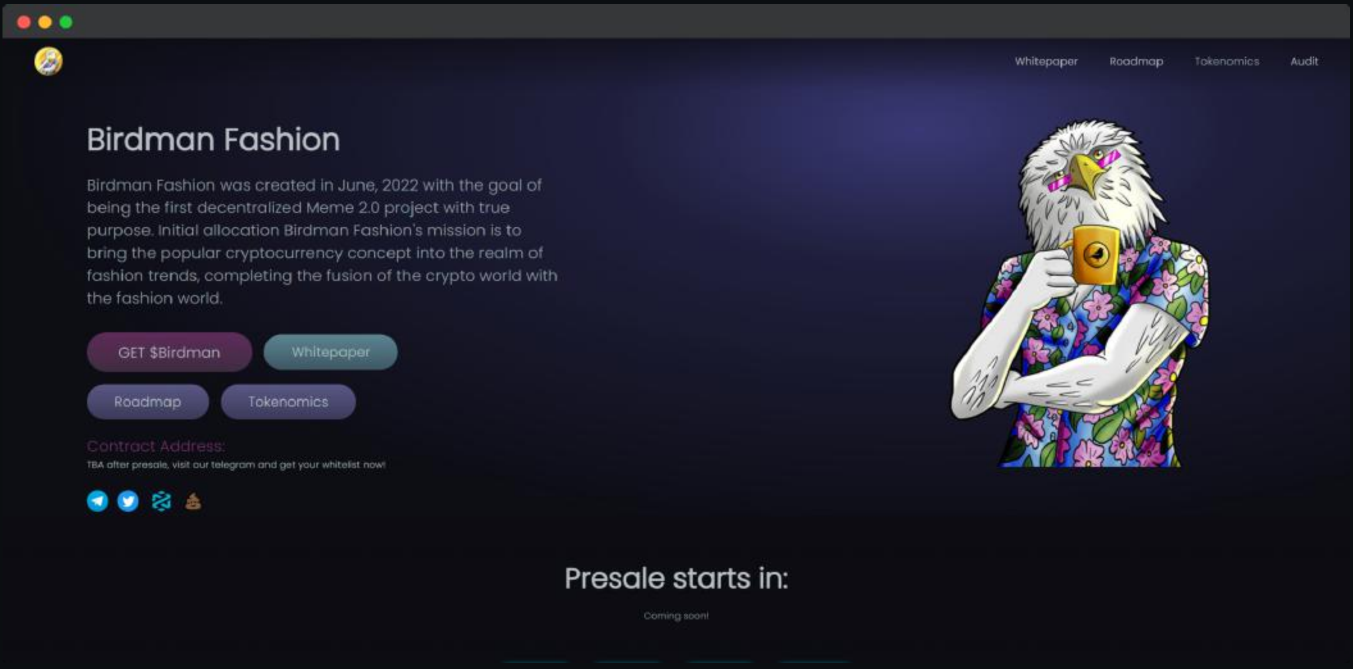
No notes

Contract Snapshot

```
contract BABYTOKEN is ERC20, Ownable {
using SafeMath for uint256;

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapPair;

bool private swapping;

BABYTOKENDividendTracker public dividendTracker;

address public rewardToken;

uint256 public swapTokensAtAmount;

uint256 public buyTokenRewardsFee;
uint256 public sellTokenRewardsFee;
uint256 public buyLiquidityFee;
uint256 public sellLiquidityFee;
uint256 public buyMarketingFee;
uint256 public sellMarketingFee;
uint256 public buyDeadFee;
uint256 public sellDeadFee;
uint256 public AmountLiquidityFee;
uint256 public AmountTokenRewardsFee;
uint256 public AmountMarketingFee;

address public _marketingWalletAddress;

address public deadWallet = 0x000000000000000000000000000000000000dEaD;

uint256 public gasForProcessing;

bool public swapAndLiquifyEnabled = true;

 // exlcude from fees and max transaction amount
mapping (address =&gt; bool) private _isExcludedFromFees;
```

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.



Birdman Fashion

Birdman Fashion was created in June, 2022 with the goal of being the first decentralized Meme 2.0 project with true purpose. Initial allocation Birdman Fashion's mission is to bring the popular cryptocurrency concept into the realm of fashion trends, completing the fusion of the crypto world with the fashion world.

Contract Address:

TBA after presale, visit our telegram and get your whitelist now!

Presale starts in:

Coming soon!

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

Project Overview

● Not KYC verified by Coinsult

Birdman Token

Audited by Coinsult.net

Date: 13 July 2022

✓ Advanced Manual Smart Contract Audit