Coinsult

Advanced Manual Smart Contract Audit

Project: Adam

Website: No website

🟢 **Low-Risk**

4 low-risk code issues found

🟡 **Medium-Risk**

1 medium-risk code issues found

🔴 **High-Risk**

0 high-risk code issues found

Contract Address

0x416C6d300a69b91c5F6A3c855C8eFe8e701c4401

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...000	50,011,860.11781695989384599	50.0119%
2	Null Address: 0x000...001	17,192,116.695700372139000055	17.1921%
3	0xbc0bc33d0db2d263afc1056db276721010c31775	15,000,000	15.0000%
4	0x0599b269315273519637363733a515c0d5d3ea8e	9,916,675	9.9167%
5	0xff4c89a5dcdfb2854a56e673695c1d71232237fd	5,784,500.86	5.7845%

Source Code

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

https://bscscan.com/address/0x416c6d300a69b91c5f6a3c855c8efe8e701c4401#code

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

```
function transferFrom(address _owner, address recipient, uint256 amount) external override returns (
    _transfer(_owner, recipient, amount);
    _approve(_owner, _msgSender(), _allowances[_owner][_msgSender()].sub(amount, "BEP20: transfer am
    return true;
}
```

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 (block.timestamp &gt; START_TIME) {
            require(endTime &gt;= block.timestamp, "Account not activated");
        }
```

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.

No zero address validation for some functions

Detect missing zero address validation.

```
function setFeeCenter(address target) override external onlyOwner {
    feeCenter = target;
    emit FeeCenterChanged(target, block.timestamp);
}
```

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.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function updateLimitTime(uint newTime) external onlyOwner() {
    START_TIME = newTime;
}
```

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.

Owner can mint new tokens

```
function _mint(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: mint to the zero address");

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);
    emit Transfer(address(0), account, amount);
}
```

Recommendation

No recommendation

Owner privileges

- 🟢 Owner cannot change max transaction amount
- 🟡 Owner can set fees higher than 25%
- 🟡 Owner can pause the contract
- 🔴 Owner can mint new tokens
- 🔴 Owner can blacklist addresses

Extra notes by the team

No notes

Contract Snapshot

```
contract AdamToken is IBEP20, FeeConfig, Ownable {

using SafeMath for uint256;

AdamAccount ADAM_ACCOUNT;

constructor(address relation){
    _decimals = 18;
    _symbol = "ADAM";
    _name = "ADAM TOKEN";
    _mint(_msgSender(), 10000 * (10 ** 4) * (10 ** _decimals));
    _burn(_msgSender(), 5000 * (10 ** 4) * (10 ** _decimals));

    _creator = _msgSender();

    ADAM_ACCOUNT = AdamAccount(relation);

    START_TIME = 7962681600;
}

address private _creator;

mapping(address =&gt; uint256) private _balances;
mapping(address =&gt; mapping(address =&gt; uint256)) private _allowances;
uint256 private _totalSupply;
string private _name;
string private _symbol;
uint8 private _decimals;
```

Project Overview

● Not KYC verified by Coinsult