Coinsult

Advanced Manual Smart Contract Audit

Project: AlchemyCrypto

Website: https://alchemycrypto.app/#/home

● Low-Risk	● Medium-Risk	● High-Risk
3 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract Address

Not yet deployed

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

Not available

Source Code

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

https://github.com/AlchemyCryptoBSC/SmartContract/blob/main/AlchemyCryptoToken.sol

While Coinsult checks the main contract for issues we can't guarantee the correctness and legitness of proxied contracts over-time. Furthermore Coinsult does not check the imports within the main contract. Always DYOR.

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

3 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.

Too many digits

Literals with many digits are difficult to read and review.

```
_totalSupply = 5000000 * (10 ** uint256(_decimals));
```

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.

No zero address validation for some functions

Detect missing zero address validation.

```
function addMinter(address minter) public onlyOwner {
    _minters[minter] = true;
}
```

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.

Divide before multiply

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

```
uint8 temp = (48 + uint8(_i - _i / 10 * 10));
```

Recommendation

Consider ordering multiplication before division.

Exploit scenario

```
contract A {
    function f(uint n) public {
        coins = (oldSupply / n) * interest;
    }
}
```

If n is greater than `oldSupply`, `coins` will be zero. For example, with `oldSupply = 5; n = 10, interest = 2`, coins will be zero. If `(oldSupply * interest / n)` was used, `coins` would have been 1. In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

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

Owner can mint new tokens

```
function mint(address _to, uint256 _amount) onlyMinter public returns (bool)    {
    uint256 tmpTotal = currentSupply + _amount;
    require(tmpTotal &lt;= _totalSupply, &quot;mint too much&quot;);
    currentSupply = currentSupply + _amount;
    balances[_to] = balances[_to] + _amount;
    emit Mint(_to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
}
```

Recommendation

No recommendation

Owner privileges

- 🟡 Owner can change max transaction amount
- 🟡 Owner can set fees higher than 25%
- 🟡 Owner can exclude from fees
- 🟡 Owner can pause the contract
- 🔴 Owner can mint new tokens

Owner can add new minter addresses

Extra notes by the team

No notes

Contract Snapshot

```
contract A_AlchemyCryptoToken is GlobalImpl, IERC20 {
IAddressManager public addressManager;
bool public pause = false;

string  _name;
string  _symbol;
uint8  _decimals;
// max supply
uint256 _totalSupply;
// current supply
uint256 public currentSupply = 0;
```

Project Overview

● Not KYC verified by Coinsult