Coinsult

Advanced Manual Smart Contract Audit

Project: MinerCity

Website: https://minercity.io/

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

Contract Address

0x64727B318Eb02630022342B700DDECa5baAA5953

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	0x1375093ac98e65783a65ea09355aca78d95b6aed	10,000,000,000	100.0000%

Source Code

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

https://bscscan.com/address/0x64727B318Eb02630022342B700DDECa5baAA5953#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

7 low-risk code
issues found

🟡 Medium-Risk

0 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

```
// Override
function _transfer(address sender, address recipient, uint256 amount) internal override {

    require(hasRole(BLACKLIST_ROLE, sender) == false &amp;&amp; hasRole(BLACKLIST_ROLE, recipient) =

    if (isWhitelisted[sender] || isWhitelisted[recipient] || inSwap) {
        super._transfer(sender, recipient, amount);
        return;
    }

    require(tradingEnabled);

    if (_shouldSwapBack(isMarketMaker[recipient])) { _swapBack(); }

    uint256 amountAfterTaxes = _takeTax(sender, recipient, amount);

    super._transfer(sender, recipient, amountAfterTaxes);

}
```

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.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled);
    tradingEnabled = true;
    launchTime = block.timestamp;
    emit EnableTrading();
}
```

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.

Too many digits

Literals with many digits are difficult to read and review.

```
function setTransferGas(uint256 newGas) external onlyOwner {
    require(newGas &gt;= 25000 &amp;&amp; newGas &lt;= 500000);
    transferGas = newGas;
    emit SetTransferGas(newGas);
}
```

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.

Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
function _swapBack() private swapping {

    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = PancakeSwapV2Router.WETH();

    uint256 liquidityTokens = swapThreshold * liquidityShare / totalShares / 2;
    uint256 amountToSwap = swapThreshold - liquidityTokens;
    uint256 balanceBefore = address(this).balance;

    PancakeSwapV2Router.swapExactTokensForETH(
        amountToSwap,
        0,
        path,
        address(this),
        block.timestamp
    );

    uint256 amountBNB = address(this).balance - balanceBefore;
    uint256 totalBNBShares = totalShares - liquidityShare / 2;

    uint256 amountBNBLiquidity = amountBNB * liquidityShare / totalBNBShares / 2;
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

Exploit scenario

```
contract ArbitrarySend{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

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

Unchecked transfer

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

```
function recoverBEP20(IERC20 token, address recipient) external onlyOwner {
    require(address(token) != address(this), "Can't withdraw");
    uint256 amount = token.balanceOf(address(this));
    token.transfer(recipient, amount);
    emit RecoverBEP20(address(token), amount);
}
```

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.

Boolean equality

Detects the comparison to boolean constants.

```
// Override
function _transfer(address sender, address recipient, uint256 amount) internal override {

    require(hasRole(BLACKLIST_ROLE, sender) == false &amp;&amp; hasRole(BLACKLIST_ROLE, recipient) =

    if (isWhitelisted[sender] || isWhitelisted[recipient] || inSwap) {
        super._transfer(sender, recipient, amount);
        return;
    }
```

Recommendation

Remove the equality to the boolean constant.

Exploit scenario

```
contract A {
    function f(bool x) public {
        // ...
        if (x == true) { // bad!
           // ...
        }
        // ...
    }
}
```

Boolean constants can be used directly and do not need to be compare to `true` or `false`.

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

Conformance to Solidity naming conventions

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
Function ERC20Permit.DOMAIN_SEPARATOR() (@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20P
Variable ERC20Permit._PERMIT_TYPEHASH_DEPRECATED_SLOT (@openzeppelin/contracts/token/ERC20/extension
Function IERC20Permit.DOMAIN_SEPARATOR() (@openzeppelin/contracts/token/ERC20/extensions/draft-IERC2
Variable EIP712._CACHED_DOMAIN_SEPARATOR (@openzeppelin/contracts/utils/cryptography/draft-EIP712.so
Variable EIP712._CACHED_CHAIN_ID (@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#32) is
Variable EIP712._CACHED_THIS (@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#33) is not
Variable EIP712._HASHED_NAME (@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#35) is not
Variable EIP712._HASHED_VERSION (@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#36) is
Variable EIP712._TYPE_HASH (@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#37) is not i
Variable MinerCity.PancakeSwapV2Router (contracts/MinerCity.sol#24) is not in mixedCase
Variable MinerCity.PancakeSwapV2Pair (contracts/MinerCity.sol#25) is not in mixedCase
Function IPancakeSwapV2Router01.WETH() (contracts/Pancakeswap.sol#7) is not in mixedCase
Function IPancakeSwapV2Pair.DOMAIN_SEPARATOR() (contracts/Pancakeswap.sol#155) is not in mixedCase
Function IPancakeSwapV2Pair.PERMIT_TYPEHASH() (contracts/Pancakeswap.sol#156) is not in mixedCase
Function IPancakeSwapV2Pair.MINIMUM_LIQUIDITY() (contracts/Pancakeswap.sol#174) is not in mixedCase
```

Recommendation

Follow the Solidity naming convention.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow _ at the beginning of the `mixed_case` match for private variables and unused parameters.

Owner privileges

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

⚠️ Trading disabled by default, owner can only set trading to enabled once.

⚠️ Owner can grant multiple roles (blacklist / admin etc.)

⚠️ If ‘sniperTax’ enabled (default true), tax will be 99%

⚠️ If ownership is rennounced, admin role can still grant roles:

```
 function grantRole(bytes32 role, address account) public virtual override onlyRole(getRoleAdmin(ro
_grantRole(role, account);
}
```

Extra notes by the team

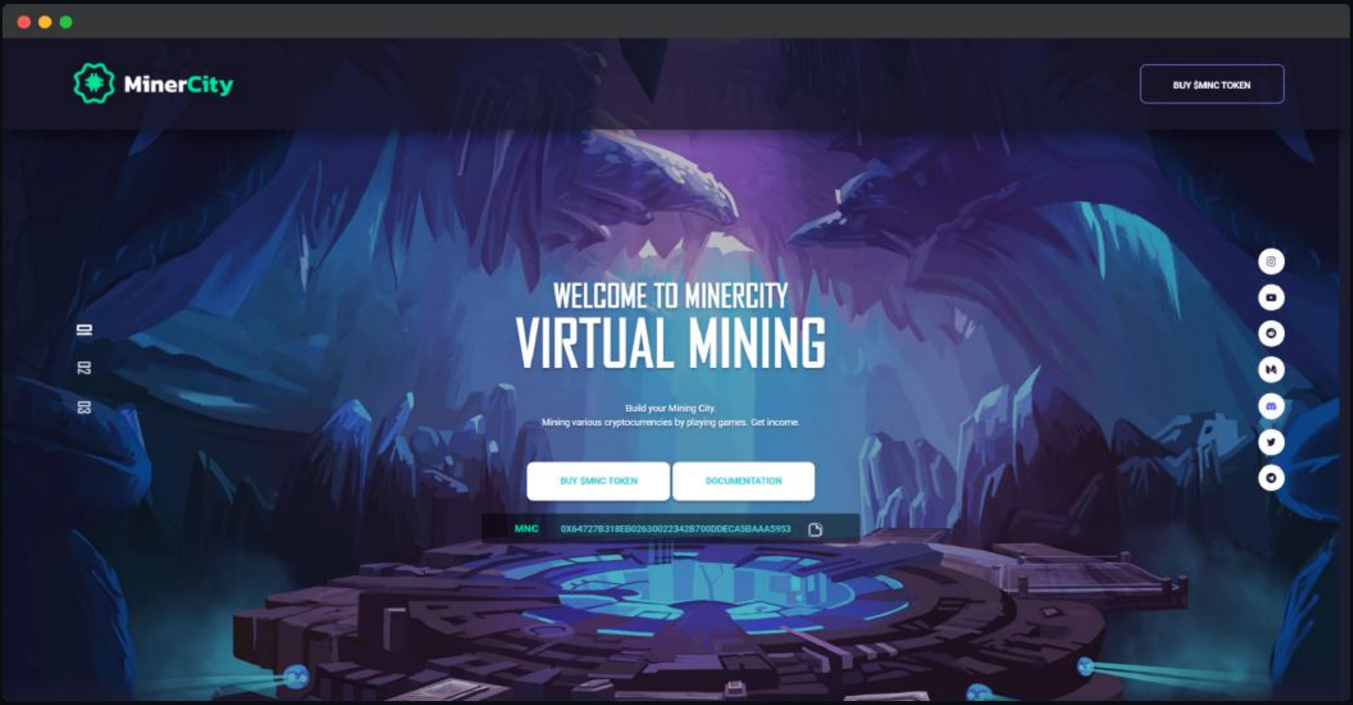
No notes

Contract Snapshot

```
contract MinerCity is
  ERC20, ERC20Permit,
  ERC20Capped, ERC20Burnable,
  Ownable, AccessControl {

using SafeMath for uint256;

IPancakeSwapV2Router02 private PancakeSwapV2Router;
address private PancakeSwapV2Pair;

bytes32 private constant BLACKLIST_ROLE = 0x0000000000000000000000000000000000000000000000000000

bool projectInitialized;

uint256 public maxSupply = 10 * 10**9 * 10 ** decimals();
uint256 private _totalSupply;
uint256 public swapThreshold;
bool public swapEnabled;

bool sniperTax;
bool tradingEnabled;
bool inSwap;

uint256 public buyTax;
uint256 public sellTax;
uint256 public transferTax;

uint256 public liquidityShare;
uint256 public marketingShare;
uint256 constant TAX_DENOMINATOR=100;
uint256 totalShares;

uint256 public transferGas;
uint256 public launchTime;

address marketingWallet;
```

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.



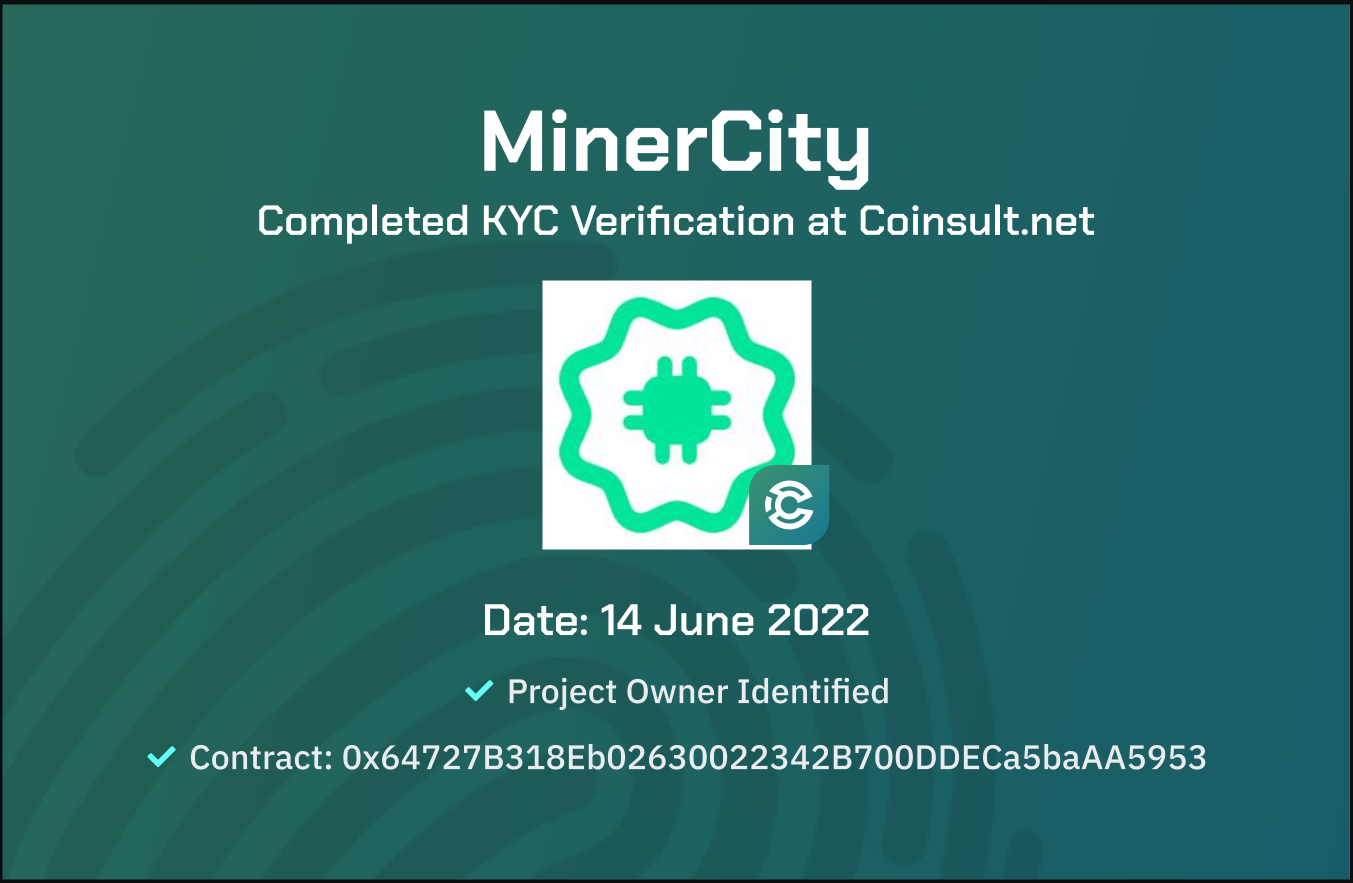
Build your Mining City.
Mining various cryptocurrencies by playing games. Get income.

MNC 0X64727B318EB026300223428700DDECA5BAAA5953

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

Project Overview

● KYC verified by Coinsult



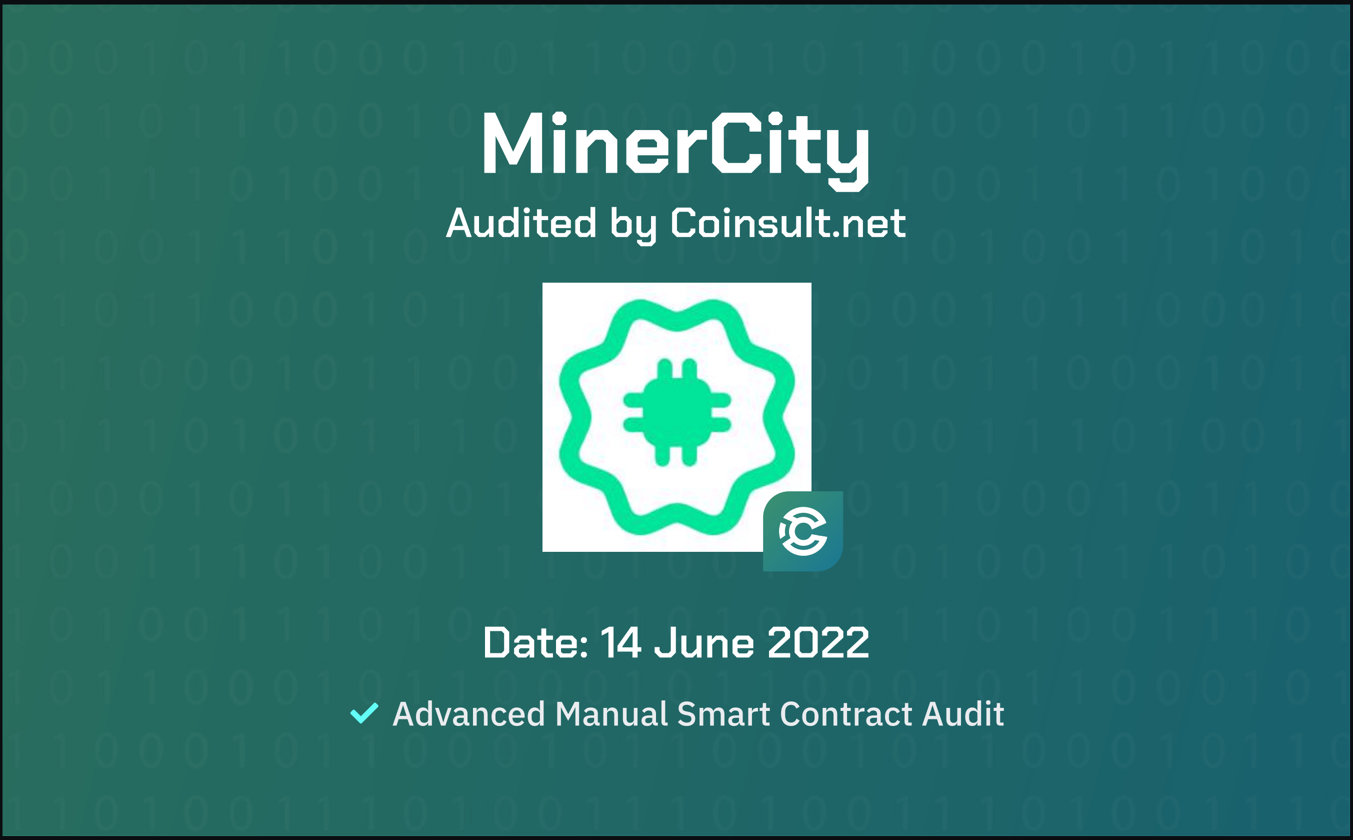
MinerCity

Completed KYC Verification at Coinsult.net

Date: 14 June 2022

✓ Project Owner Identified

✓ Contract: 0x64727B318Eb02630022342B700DDECa5baAA5953



MinerCity

Audited by Coinsult.net

Date: 14 June 2022

✓ Advanced Manual Smart Contract Audit