

# Advanced Manual **Smart Contract Audit**

September 26, 2022

Audit requested by



**RunEarner**

0x82d2c560b9aFfFf5049bE54536C922A462D717e4

# Table of Contents

## 1. Audit Summary

- 1.1 Audit scope
- 1.2 Tokenomics
- 1.3 Source Code

## 2. Disclaimer

## 3. Global Overview

- 3.1 Informational issues
- 3.2 Low-risk issues
- 3.3 Medium-risk issues
- 3.4 High-risk issues

## 4. Vulnerabilities Findings

## 5. Contract Privileges

- 5.1 Maximum Fee Limit Check
- 5.2 Contract Pausability Check
- 5.3 Max Transaction Amount Check
- 5.4 Exclude From Fees Check
- 5.5 Ability to Mint Check
- 5.6 Ability to Blacklist Check
- 5.7 Owner Privileges Check

## 6. Notes

- 6.1 Notes by Coinsult
- 6.2 Notes by RunEarner

## 7. Contract Snapshot

## 8. Website Review

## 9. Certificate of Proof

# Audit Summary

## Audit Scope

Project Name	RunEarner
Website	<a href="https://www.runearder.com/">https://www.runearder.com/</a>
Blockchain	Binance Smart Chain
Smart Contract Language	Solidity
Contract Address	0x82d2c560b9aFfF5049bE54536C922A462D717e4
Audit Method	Static Analysis, Manual Review
Date of Audit	26 September 2022

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

## Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0x1c5b16c2ca4677f13a2d8128e95f96f9fd8f84ab	500,000	100.0000%

## Source Code

Coinsult was commissioned by RunEarner to perform an audit based on the following code:

<https://bscscan.com/address/0x82d2c560b9aFfFf5049bE54536C922A462D717e4#code>

**Safu by ContractChecker**

# Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

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.

# Global Overview







## Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
 Informational	0	0	0	0
 Low-Risk	5	5	0	0
 Medium-Risk	0	0	0	0
 High-Risk	0	0	0	0

## Privilege Overview

Coinsult checked the following privileges:

Contract Privilege	Description
Owner can mint?	 Owner cannot mint new tokens
Owner can blacklist?	 Owner cannot blacklist addresses
Owner can set fees > 25%?	 Owner cannot set the sell fee to 25% or higher
Owner can exclude from fees?	 Owner can exclude from fees
Owner can pause trading?	 Owner cannot pause the contract
Owner can set Max TX amount?	 Owner cannot set max transaction amount

More owner privileges are listed later in the report.

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

## Mixed up parameters in UpdateBuyFees event

Event:

```
UpdateBuyFees(uint256 liquidityFeeonBuy, uint256 treasuryFeeonBuy, uint256 rewardFeeonBuy, uint256 rewardFeeonBuy)
```

Emitted event:

```
emit UpdateBuyFees(_liquidityFeeonBuy, _treasuryFeeonBuy, _insuranceFeeonBuy, _rewardFeeonBuy, _rewardFeeonBuy)
```

## Recommendation

Change the order of the events for readability and traceability.

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

## Too many digits

Literals with many digits are difficult to read and review.

```
function setSwapTokensAtAmount(bool _swapEnabled,uint256 newAmount) external onlyOwner{
    require(newAmount > rSupply / 100000, "SwapTokensAtAmount must be greater than 0.001% of total supply");
    swapThreshold = newAmount;
    swapEnabled = _swapEnabled;
    emit SwapTokensAtAmountUpdated(_swapEnabled,newAmount);
}

function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue >= 200000 && newValue <= 800000, "gasForProcessing must be between 200,000 and 800,000");
    require(newValue != gasForProcessing, "Cannot update gasForProcessing to same value");
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

## Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

## Exploit scenario

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

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 changeTreasuryWallet(address _treasuryWallet) external onlyOwner {
    require(_treasuryWallet != treasuryWallet, "Treasury wallet is already that address");
    require(!isContract(_treasuryWallet), "Treasury wallet cannot be a contract");
    treasuryWallet = _treasuryWallet;
    emit TreasuryWalletChanged(treasuryWallet);
}

function changeInsuranceWallet(address _insuranceWallet) external onlyOwner {
    require(_insuranceWallet != insuranceWallet, "Insurance wallet is already that address");
    require(!isContract(_insuranceWallet), "Insurance wallet cannot be a contract");
    insuranceWallet = _insuranceWallet;
    emit InsuranceWalletChanged(insuranceWallet);
}
```

## 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, so Bob 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 claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

## 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 updateMinimumBalanceForDividends(uint256 newMinimumBalance) external onlyOwner {
    dividendTracker.updateMinimumTokenBalanceForDividends(newMinimumBalance);
}

function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

## 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.

# Contract Privileges

## Maximum Fee Limit Check


Coinsult tests if the owner of the smart contract can set the transfer, buy or sell fee to 25% or more. It is bad practice to set the fees to 25% or more, because owners can prevent healthy trading or even stop trading when the fees are set too high.

Type of fee	Description
Transfer fee	● Owner cannot set the transfer fee to 25% or higher
Buy fee	● Owner cannot set the buy fee to 25% or higher
Sell fee	● Owner cannot set the sell fee to 25% or higher

Type of fee	Description
Max transfer fee	0%
Max buy fee	25%
Max sell fee	25%

## Contract Pausability Check

Coinsult tests if the owner of the smart contract has the ability to pause the contract. If this is the case, users can no longer interact with the smart contract; users can no longer trade the token.

Privilege Check	Description
Can owner pause the contract?	 Owner cannot pause the contract


## Max Transaction Amount Check

Coinsult tests if the owner of the smart contract can set the maximum amount of a transaction. If the transaction exceeds this limit, the transaction will revert. Owners could prevent normal transactions to take place if they abuse this function.

Privilege Check	Description
Can owner set max tx amount?	● Owner cannot set max transaction amount

## Exclude From Fees Check

Coinsult tests if the owner of the smart contract can exclude addresses from paying tax fees. If the owner of the smart contract can exclude from fees, they could set high tax fees and exclude themselves from fees and benefit from 0% trading fees. However, some smart contracts require this function to exclude routers, dex, cex or other contracts / wallets from fees.


Privilege Check	Description
Can owner exclude from fees?	 Owner can exclude from fees

## Ability To Mint Check

Coinsult tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-fixed, allowing the token owner to "mint" more tokens whenever they want.

A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.

Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.


Privilege Check	Description
Can owner mint?	 Owner cannot mint new tokens



## Ability To Blacklist Check

Coinsult tests if the owner of the smart contract can blacklist accounts from interacting with the smart contract. Blacklisting methods allow the contract owner to enter wallet addresses which are not allowed to interact with the smart contract.

This method can be abused by token owners to prevent certain / all holders from trading the token. However, blacklists might be good for tokens that want to rule out certain addresses from interacting with a smart contract.

Privilege Check	Description
Can owner blacklist?	 Owner cannot blacklist addresses

## Other Owner Privileges Check

Coinsult lists all important contract methods which the owner can interact with.

⚠ Owner can exclude addresses from dividend

# Notes

## Notes by RunEarner

No notes provided by the team.

## Notes by Coinconsult

 No notes provided by Coinconsult

# Contract Snapshot

This is how the constructor of the contract looked at the time of auditing the smart contract.

```
contract RunEarner is ERC20, Ownable {
    mapping (address => uint256) _rBalance;
    mapping (address => mapping(address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFees;

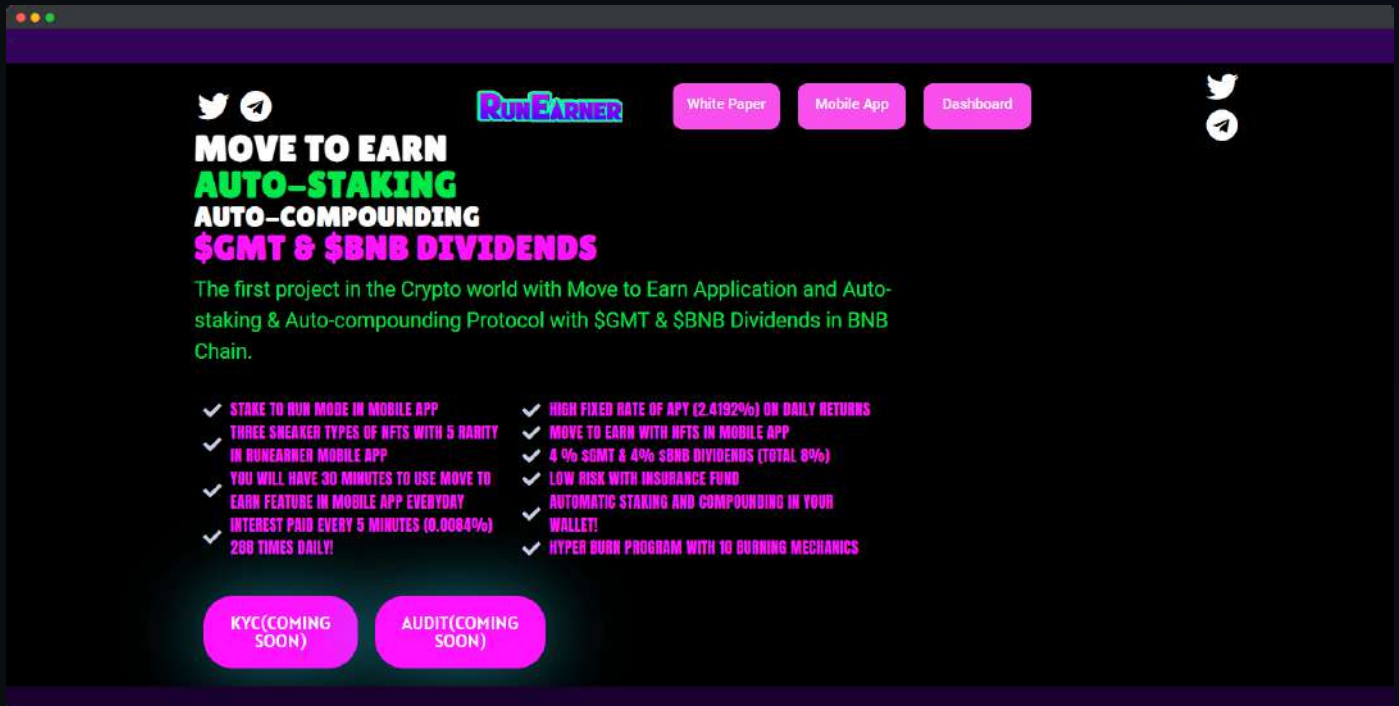
    uint256 public liquidityFeeonBuy;
    uint256 public liquidityFeeonSell;

    uint256 public treasuryFeeonBuy;
    uint256 public treasuryFeeonSell;

    uint256 public rewardFeeonBuy;
    uint256 public rewardFeeonSell;
```

# 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.



Type of check	Description
Mobile friendly?	● The website is mobile friendly
Contains jQuery errors?	● The website does not contain jQuery errors
Is SSL secured?	● The website is SSL secured
Contains spelling errors?	● The website does not contain spelling errors

# Certificate of Proof

● Not KYC verified by Coinsult

## RunEarner

Audited by Coinsult.net



Date: 26 September 2022

✓ Advanced Manual Smart Contract Audit

End of report  
**Smart Contract Audit**

Request your smart contract audit / KYC

**[t.me/coinsult\\_tg](https://t.me/coinsult_tg)**