



BULLS PROTOCOL

Smart Contract security and compliance assessments

CupBoll.bet



BullsProtocol.com

December 25, 2022

INTRODUCTION

This audit has resulted from Smart Contract analysis and testing against common and uncommon vectors used in code attacks, analyzes from a cyber vulnerabilities perspective, and also whether the code structure follows best practices in industry standards.

The works are developed through some methodologies, among them the revision of all lines of the source code.

Reviews result in ratings ranging from critical to approved. A rating is given as **CRITICAL** if it allows for potential exploitation of the problem by attackers, if the manifestation of the bug causes financial loss or if it interrupts the execution of the code.

After the **CRITICAL** classification we also have the **HIGH**, **MEDIUM** and **LOW** classifications.

LOW severity ratings indicate points that do not compromise the code or its functioning.

The next classification would be the **INFORMATIVE**, these classifications are just simple indications and comments about the source code.

And finally, the **APPROVED**, which are all those that passed all analyzes and do not fit in the previous cases.

The indications of **NOT FOUND** refer to situations not found in the contract, which also mean that they have the status of **APPROVED**

ÍTALO HONORATO
BLOCKCHAIN AUDITOR

PEDRO CHALEGRE
BUSINESS SPECIALIST



OVERVIEW

The contract has only one file, **CupBoll.sol**, which has a source code with 810 lines. We verified that all functions and variables were used and commented in accordance with the Ethereum EPI documentation and standards, the OpenZeppelin standard libraries and also in accordance with other technical standards.

Project name	CupBoll Bolão
Source Code	https://bscscan.com/token/0x5DB4A5616c21F75E71D4399F8e71D621e8de3fdb#code
Contract address	0x5DB4A5616c21F75E71D4399F8e71D621e8de3fdb
ByteCode for Ethereum Virtual Machine	https://bscscan.com/bytecode-decompiler?a=0x5DB4A5616c21F75E71D4399F8e71D621e8de3fdb
Contract fees	10% buy and 12% sell
Liquidity fees	5% buy and 5% sell
Fees for development wallet	6% buy and 6% sell
Website	http://cupboll.bet/
Telegram	https://t.me/Cupbollbolao



CHECKING STATUS

Table of audit analysis results:

Code accuracy according to expectations	APPROVED
Errors when compiling	APPROVED
Timestamp dependency	APPROVED
Code testability	APPROVED
Clean/lean code	APPROVED
DDOS attack by gas limit per block	APPROVED
DDOS attack due to failed function calls	APPROVED
DDOS attack by reversing transaction processing	APPROVED
Use of outdated functions in Solidity	APPROVED
Reentry attack/reentrancy attack	APPROVED
Poorly documented or unprotected SELFDESTRUCT statements	NOT FOUND
Not checking return value in Get functions	APPROVED
Integer overflow/underflow	APPROVED
Presence of unused variables	APPROVED
Code redundancy	APPROVED
Gas optimization in transactions	APPROVED
Optimization of contract execution	APPROVED
Prevention of loss of values within the contract	APPROVED
Violation of REQUIRE FUCTION	APPROVED
Typographical errors	APPROVED
Unsecured BNB withdrawal	APPROVED
Logical design	APPROVED
Arithmetic precision	APPROVED
Fallback function security	APPROVED
Compliance with EIP standards	APPROVED
Disclaimer of ownership	APPROVED
Implementation of standard and secure OpenZeppelin libraries	APPROVED



PRIVILEGED FUNCTIONS

The contract has functions that can only be modified by the owner of the contract, owner defined by the ***onlyOwner*** or ***OnlyAuth*** modifier.

Privileged roles have exclusive access to:

- ***updateUniswapV2Router*** (storage in type address 20 bytes), changes the Pancake V2 route address;
- ***airdrop***, distributes airdrop tokens;
- ***setNoAllowedToTrade***, add or remove blocklist accounts;
- ***setMappingAuth***, used by the owner to add or remove addresses from the admin list;
- ***setB_Swap***, changes buy rates;
- ***setS_Swap***, changes sell rates;
- ***setProjectWallet***, change the address of the marketing wallet;
- ***burnToZeroAddress***, used to burn tokens to address zero;
- ***burnOfSupply***, used to burn supply tokens;
- ***setTriggerSwapTokensToBNB***, sets the trigger point between exchanging tokens for BNB;
- ***setWhatsSwapPriceBurn***, used to define which of the previous functions are used in automatic token burning;
- ***swapGrowthPrice***, buys tokens for price increase and immediately burns those tokens.



ControlledSwap CONTRACT

It is verified that the main contract created and controls a child contract. Such a secondary contract is used to store funds and to route funds transfers.

The justification used in the comments is that PancakeSwap's code architecture does not facilitate the control of funds without using the secondary contract.

```
//This auxiliary contract is necessary for the logic of the liquidity mechanism to work
//The pancake router V2 does not allow the address(this) to be
//in swap and at the same time be the destination of "to"
//This contract is where the funds will be stored
//The movement of these funds (CBL and BNB) is done exclusively by the token's main contract
contract ControlledSwap is Ownable {
```



COMMENTS IN THE CODE

Codes are punctually commented when more complex functions or mechanisms are approached. Below are two examples of developer clarifications. In the first image there is a justification about the unnecessary use of SafeMath libraries, with gas savings being one of the reasons:

```
uint256 senderBalance = _balances[from];
require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
//Solidity above version 0.8.0 makes safe math calculations for all operations
//The instruction not to check below helps save gas that would be spent unnecessarily
unchecked {
    _balances[from] = senderBalance - amount;
    //When calculating fees, it is always guaranteed that amount > fees
    _balances[to] += (amount - fees);
    _balances[address(controlledSwap)] += fees;
    amount = amount - fees;
}
```

```
//buy tokens
if (automatedMarketMakerPairs[from]) {
    /* Multiplication never results in an overflow because
       variable entries are in expected interval.
       Amount and fees are within defined interval, never under or over
       That is, fees <= amount * totalBuy / 10000 always
    */
    unchecked {fees = amount * totalBuy / 10000;}
```



RUG-PULL FUNCTIONS

There are no functions or mechanisms in the contract that can be used by the owner to perform rug-pulls. The following functions or logics were not found:

- deliberate token creation schemes (*mint* function)
- functions to lock trades
- contract self-destruct mechanisms (*selfdestruct* function) to make it difficult to trace transactions

The trade once activated can never be deactivated:

```
//Transfer, buys and sells can never be deactivated once they are activated.
/*The name of this function is due to bots and automated token
parsing sites that parse only by name but not by function
and always come to incorrect conclusions when they say that this function can be disabled
*/
function onlyActivatedNeverOff() external onlyOwner() {
    require(alwaysOnNeverOff[address(this)] == false, "Already open");

    timeLaunched = block.timestamp;
    alwaysOnNeverOff[address(this)] = true;

    emit launchEvent(timeLaunched, true);
}
```



CONCLUSION

The audit performed manual line-by-line analysis and automated review of the smart contract. The contract and source code were analyzed mainly from the point of view of common vulnerabilities, exploits, manipulation hacks and code structure optimizations.

The contract and source code have an **APPROVED** status in the audit reviews and discussions.

No issues were found that fall under the status of **CRITICAL**, **HIGH**, **MEDIUM**, or **LOW**.



LEGAL NOTICE

BULLS PROTOCOL and its technical team provide through this audit report only discussion and evaluation about project and contract, with the sole intention of helping to improve the quality of the code and to improve the level of variation used in smart contract technologies. Therefore, this audit report is not a legally binding document.

This technical audit report is not a buy or sell recommendation, approval or disapproval, or even a definitive answer on the points discussed here, but only expresses the technical and analytical opinion on the **CupBoll Bolão** project.

The reproduction of this material is free and there is no need to request permission from the creators, but the source must be cited.

