

Audit report of MBP COIN

Prepared By: - Kishan Patel
Prepared On: - 24/04/2024.

Prepared for: MBP COIN

Table of contents

- 1. Disclaimer**
- 2. Introduction**
- 3. Project information**
- 4. List of attacks checked**
- 5. Severity Definitions**
- 6. Good things in code**
- 7. Critical vulnerabilities in code**
- 8. Medium vulnerabilities in code**
- 9. Low vulnerabilities in code**
- 10. Summary**

THIS AUDIT REPORT WILL CONTAIN CONFIDENTIAL INFORMATION ABOUT THE SMART CONTRACT AND INTELLECTUAL PROPERTY OF THE CUSTOMER AS WELL AS INFORMATION ABOUT POTENTIAL VULNERABILITIES OF THEIR EXPLOITATION.

THE INFORMATION FROM THIS AUDIT REPORT CAN BE USED INTERNALLY BY THE CUSTOMER OR IT CAN BE DISCLOSED PUBLICLY AFTER ALL VULNERABILITIES ARE FIXED - UPON THE DECISION OF THE CUSTOMER.

1. Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions). Because the total numbers of test cases are unlimited, the audit makes no statements or warranties on the security of the code.

It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

2. Introduction

Kishan Patel (Consultant) was contacted by MBP COIN. (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contracts and its code review conducted between 24/04/2024 – 26/04/2024.

The project has 1 file. It contains approx 500 lines of Solidity code. All the functions and state variables are well commented on using the natspec documentation, but that does not create any vulnerability.

3. Project information

Token Name	MBP COIN
Token Symbol	MBP
Platform	Bscscan
Order Started Date	24/04/2024
Order Completed Date	26/04/2024

4. List of attacks checked

- Over and under flows
- Short address attack
- Visibility & Delegate call
- Reentrancy / TheDAO hack
- Forcing BNB to a contract
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference

5. Severity Definitions

Risk	Level Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution

6. Good things in code

- **Good required condition in functions:-**

- Here smart contract is checking that newOwner address is valid and proper.

```
336     */
337     function _transferOwnership(address newOwner) internal {
338         require(newOwner != address(0), "Ownable: new owner is the zero address");
339         emit OwnershipTransferred(_owner, newOwner);
340         _owner = newOwner;
```

- Here smart contract is checking that sender and recipient addresses are valid and proper.

```
536     function _transfer(address sender, address recipient, uint256 amount)
537         require(sender != address(0), "BEP20: transfer from the zero address");
538         require(recipient != address(0), "BEP20: transfer to the zero address");
```

- Here smart contract is checking that account address is valid and proper.

```
556     */
557     function _burn(address account, uint256 amount) internal {
558         require(account != address(0), "BEP20: burn from the zero address");
559
```

- Here smart contract is checking that owner and spender addresses are valid and proper.

```
577     */
578     function _approve(address owner, address spender, uint256 amount) internal {
579         require(owner != address(0), "BEP20: approve from the zero address");
580         require(spender != address(0), "BEP20: approve to the zero address");
581
```


7. Critical vulnerabilities in code

- No Critical vulnerabilities found

8. Medium vulnerabilities in code

- No Medium vulnerabilities found

9. Low vulnerabilities in code

9.1. Suggestions to add code validations:-

=> You have implemented required validation in contract.

=> There are some place where you can improve validation and security of your code.

=> These are all just suggestion it is not bug.

○ Function: - `_approve`

```
577  
578     function _approve(address owner, address spender, uint256 amount) in  
579         require(owner != address(0), "BEP20: approve from the zero address  
580         require(spender != address(0), "BEP20: approve to the zero address  
581  
582         _allowances[owner][spender] = amount;  
583         emit Approval(owner, spender, amount);  
584     }
```

- Here in `_approve` function smart contract can check that owner address has sufficient balance to give allowance to spender address.

10. Summary

- **Number of problems in the smart contract as per severity level**

Critical	Medium	Low
0	0	1

According to the assessment, the smart contract code is well secured. The code is written with all validation and all security is implemented. Code is performing well and there is no way to steal funds from this contract.

- **Good Point:** Code performance and quality are good. All kind of necessary validation added into smart contract and all validations are working as expected.
- **Suggestions:** Please try to implement suggested code validations.