Audit report of NCoin

Prepared By: - Kishan Patel
Prepared On: - 21/08/2023.

Prepared for: NCoin

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 Natronz. (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 21/08/2023 – 23/08/2023.

The project has 1 file. It contains approx 400 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	NCoin
Token Symbol	NCoin
Platform	Binance Smart Contract
Order Started Date	21/08/2023
Order Completed Date	23/08/2023

4. List of attacks checked

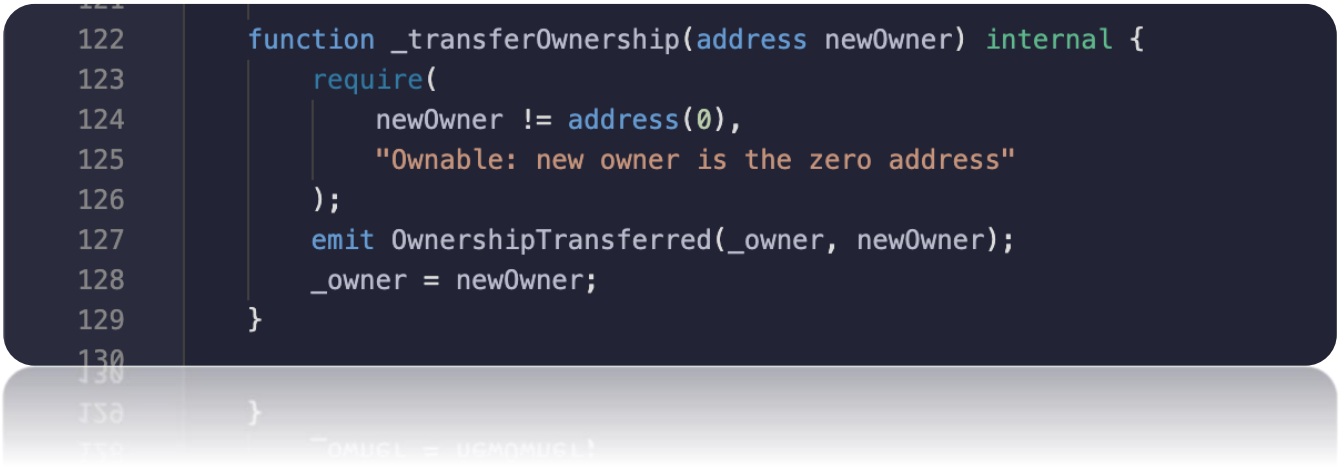
- Over and under flows
- Short address attack
- Visibility & Delegate call
- Reentrancy / TheDAO hack
- Forcing BNB to a contract
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style 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

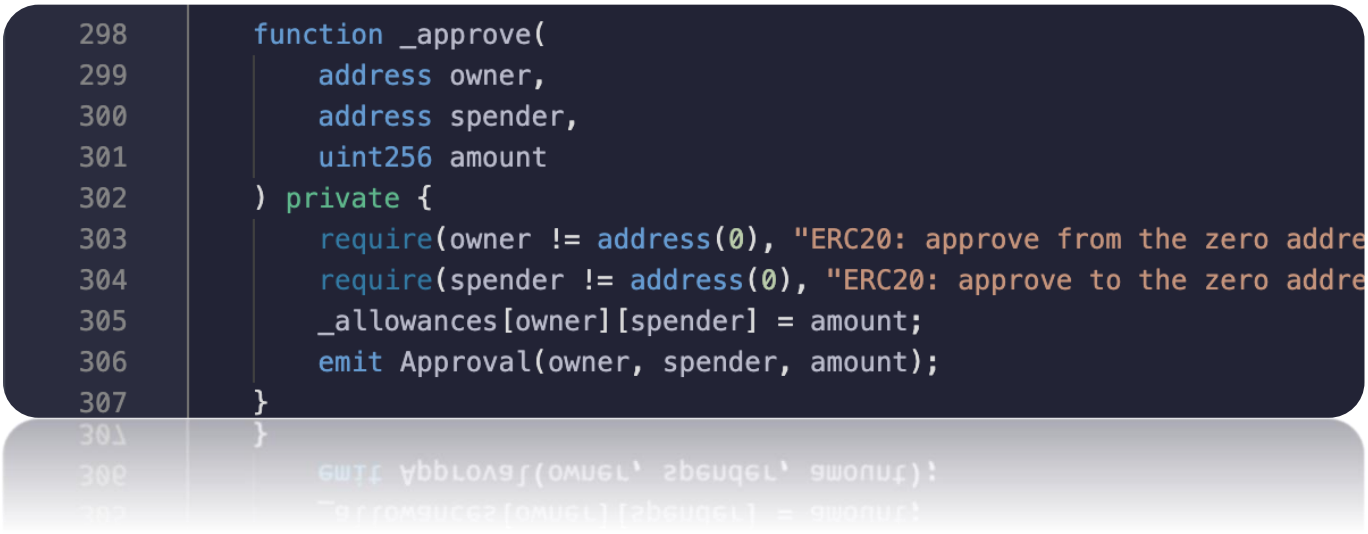
6. Good things in code

- **Good required condition in functions:-**
  - Here you are checking that newOwner address is valid and proper.



```
122     function _transferOwnership(address newOwner) internal {
123         require(
124             newOwner != address(0),
125             "Ownable: new owner is the zero address"
126         );
127         emit OwnershipTransferred(_owner, newOwner);
128         _owner = newOwner;
129     }
130
```

  - Here you are checking that owner and spender address are valid and proper.



```
298     function _approve(
299         address owner,
300         address spender,
301         uint256 amount
302     ) private {
303         require(owner != address(0), "ERC20: approve from the zero addre
304         require(spender != address(0), "ERC20: approve to the zero addre
305         _allowances[owner][spender] = amount;
306         emit Approval(owner, spender, amount);
307     }
```

- Here you are checking that from address is valid and proper, amount is bigger than min amount, trading is enabled, from and to address are whitelisted, and from and to are not blacklisted.

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**

```
298     function _approve(
299         address owner,
300         address spender,
301         uint256 amount
302     ) private {
303         require(owner != address(0), "ERC20: approve from the zero addre
304         require(spender != address(0), "ERC20: approve to the zero addre
305         _allowances[owner][spender] = amount;
306         emit Approval(owner, spender, amount);
307     }
```

- Here in approve function you can check that account owner has sufficient balance for giving allowance.

- **Function: - ChangeTax**

```
317     function ChangeTax(uint256 newBuyTax, uint256 newSellTax) external o
318         buyTax = newBuyTax;
319         sellTax = newSellTax;
320     }
```

  - Here in ChangeTax we can check that buyTax, and sellTax are bigger than 0.

- **Function: - ChangeMinSwap**

```
322     function ChangeMinSwap(uint256 NewMinSwapAmount) external onlyOwner
323         minSwap = NewMinSwapAmount;
324     }
```

  - Here in ChangeMinSwap we can check that NewMinSwapAmount ist bigger than 0.

10. Summary

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

Critical	Medium	Low
0	**0**	**3**

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 excepted.

- **Suggestions:** Please try to implement suggested code validations.