Blockchain
Contract Audit

區塊鏈合約檢測服務

Project:  AIYEEEEEE

Report date: May 31st, 2023

- ✓ Re-entrancy
- ✓ Overflow/underflow
- ✗ Use of block.timestamp
- ✓ Use of tx.origin
- ✓ Use of selfdestruct
- ✓ Storage conflict
- ✓ Force receive  token
- ✓ Using inline assembly
- ✓ Access vulnerability
- ✓ Return value of low level call
- ✓ Return value of transfer

| | |
|---|---|
| Project name | AIYEEEEEE |
| Network | BSC |
| Language | Solidity |
| Delivery Date | 2023/5 |
| Contract Address | 0x83BC666f84ca49C3c682ffAaAcCfCc4Cd5bC4b36 |

This audit report was summarised the smart contract verification service. The goal of this security audit is to guarantee that the smart contracts are perfect enough to avoid potential security vulnerability.

| | Token Information |
|---|---|
| Fee | Yes |
| Fee Privilege | Yes |
| Ownership | Yes |
| Max Tx Amount | None |
| Blacklist | None |
| Decimals | 6 |
| Max Supply | One Trillion |
| Mint/Burn | Yes |

# Re-entrancy

If a contract has this vulnerability, when it calls an external contract, and does not update its status before sending funds, an attacker could continually call the withdraw function to transfer funds until all funds in the contract are depleted.

**PASS**

# Overflow/underflow

When performing calculations on numbers, if the result exceeds or falls below the range of the type, an Overflow or Underflow vulnerability can occur.

**PASS**

# Dependance on block.timestamp

Generating random numbers using global variables like timestamp can be predicted by attackers.

**FAIL** andyToken.sol: 293

# Use of tx.origin

When a contract uses tx.origin to verify user identity, malicious actors can exploit this vulnerability, masquerading as an address that can pass verification.

**NONE**

# Use of selfdestruct

When a contract improperly uses the selfdestruct function, it can result in the contract being destroyed and its balance transferred to an address controlled by the attacker.

**NONE**

# Storage conflict

If different variables share the same storage slot, it can lead to variables being maliciously altered by attacker.

**PASS**

# Force receive token

If the balance of the contract is used as a check condition, the contract may become invalid if an attacker forces a transfer.

**PASS**

# Using inline assembly

The use of assembly is error-prone and should be avoided.

**NONE**

# Access vulnerability

Vulnerabilities in permissions may allow malicious actors to bypass identity checks for accessing functions, or to change the owner of the permissions.

**PASS**

# Return value of low level call

This vulnerability refers to an issue where, during the execution of call(), a return value is typically given to indicate whether the function was successful or not. If this return value is not properly used, unexpected errors may occur.

**NONE**

# Return value of transfer

This vulnerability refers to an issue where, during the execution of transfer(), a return value is typically given to indicate whether the transfer was successful or not. If this return value is not properly used, unexpected errors may occur.

**NONE**

# Conclusion

On-chain random numbers can be predictable or manipulatable. It is recommended to use random numbers generated by third-party sources.

## Audit Status: FAIL

# Disclaimer

Before you use this website to fill in basic information, upload information and apply to this service, you have to read this Terms of Service on the website thoroughly to protect your right.
We only audit common hacking issues in the above smart contracts, and do not guarantee the business model of this project. Investment involves risks, please consider carefully before purchasing.

Blockchain
Contract Audit