QuillAudits

Audit Report
April, 2023

For

BabyDoge

Table of Content

Executive Summary

Project Name	BabyDogeCoin Report
Overview	BabyDogeCoin ChessBetting contract is intended to provide rewards to winners ascertained from an off-chain game. The contract inherits EIP 712 and Ownable contracts from the Openzeppelin library. These contracts aid in identifying the owner of the contract and the winner of the game through their hashed signature. The manager identifies when a winner signs the withdrawal signature and then the manager sends the reward to the correct winner.
Timeline	18 July, 2022 - 9 August, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse BabyDogeCoin Chessbetting codebase for quality, security, and correctness. *https://bscscan.com/address/0x48ff73efB7dE5F760825cC3399A1F8096aEf72B8#code*
Extra Review	*Extra Review Report here.* Timeline: 31st March 2023 - 5th April 2023
Fixed In	*https://github.com/Baby-doge/BabyDogeChess-Contracts/commit/b53ce584b5e7862a0c016aa6c9f777afe734afaf*

3
Issues Found

🟥 High　🟨 Medium

🟩 Low　🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	**2**	**1**

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ DoS with Block Gas Limit
- ✓ Transaction-Ordering Dependence
- ✓ Use of tx.origin
- ✓ Exception disorder
- ✓ Gasless send
- ✓ Balance equality
- ✓ Byte array
- ✓ Transfer forwards all gas
- ✓ BEP20 API violation
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Redundant fallback function
- ✓ Send instead of transfer
- ✓ Style guide violation
- ✓ Unchecked external call
- ✓ Unchecked math
- ✓ Unsafe type inference
- ✓ Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

1. Ownership Transfer must be a Two-step Processes

Description

The Chessbetting contract inherits the Openzeppelin Ownable contract that manages the privileges allotted to the owner. While the transferOwner function checks that any passed address is not address zero, it does not completely prevent possible lapses that could come from mistakes when assigning addresses. When a wrong address is passed in as the new owner, the original owner does not have the privileges anymore to effect any change when the transferOwnership transaction is completed. This is a transfer of ownership to a wrong address. Moreso, if newOwner calls the function with the current admin address being the original owner, this is gas wastage as the contract does not prevent setting an address twice as the owner.

Recommendation

The ownership transition would be much safer by implementing a two-step mechanism. This will involve the _transferOwnership() and an _updateOwnership(). The _transferOwnership function should allow for the transfer of ownership through setting the new address in the _newOwner storage while the _updateOwnership function checks if the _newOwner is the msg.sender, when the new owner signs the transaction and verifies himself as the owner, then the _owner is set to newOwner instead of immediately changing the owner as implemented by Openzeppelin.

Status

Resolved

2. Missing Zero Address Check

Description

Contract allows for the setting of address zero as the manager of the contract. While it is impossible for a bot to sign transactions on behalf of a zero address manager, it will disrupt other critical functions and will fail when it is tried to be called by other addresses other than the address zero.

Recommendation

the setManager function should have an extra check to prevent the assignment of the zero address as the manager.

Status

Resolved

Informational Issues

3. Unlocked pragma (pragma solidity ^0.8.13)

Description

Contract has a floating solidity pragma version. This is present also in inherited contracts. Locking the pragma helps to ensure that the contract does not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. Recent solidity pragma version also possesses its own unique bugs.

Recommendation

Making the contract use a stable solidity pragma version prevents bugs occurence that could be ushered in by prospective versions. It is recommended, therefore, to use a fixed solidity pragma version while deploying to avoid deployment with versions that could expose the contract to attack.

Status

Resolved

4. General Recommendation

Chessbetting contract uses a floating pragma version that could be susceptible to attack if it is deployed by newer versions with new bugs. It is recommended to use a fixed solidity version instead to avert bug exploits. There should be additional checks for the zero address being set as the manager, to avoid failed contract calls when functions are called by the manager. It is also recommended to employ extra mechanisms for ownership transfer. This will aid in preventing erroneous transition of ownership.

NOTE: Due to the employed mechanism for the execution and signing of transactions with the assistance of bots and servers, there are possibilities of users' funds getting stuck when the system halts. This prompted adding a function called emergencyWithdraw that allows users to withdraw their funds; the contract owner must set the emergencyWithdrawalsAllowed to true to allow this function call.

Functional Testing

Some of the tests performed are mentioned below:

- ✓ should get the token data of added
- ✓ should emit data of removed token
- ✓ should revert deposit with wrong token
- ✓ should allow new token in the chessbetting
- ✓ should allow user deposit for allowed tokens
- ✓ should revert if users have insufficient funds
- ✓ should revert if wrong address set manager role
- ✓ should revert if manager already exist
- ✓ should revert calling function with manager role
- ✓ should revert withdrawal for wrong account
- ✓ should revert withdrawal for invalid token account
- ✓ should revert withdrawal for outdated deadline
- ✓ should revert withdrawal for invalid amount
- ✓ should revert repeated withdrawal
- ✓ should revert setting winner with bad signature
- ✓ should revert setting winner with 2nd bad signature
- ✓ should revert setting winner with different game id
- ✓ should revert setting winner with different bet amount
- ✓ should revert setting winner with invalid token
- ✓ should revert setting winner with invalid winner
- ✓ should revert setting winner with too big bet
- ✓ should revert repeated setting winner for the same game
- ✓ Should get the withdrawal hash
- ✓ Should get the game hash

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the BabyDogeCoin ChessBetting. We performed our audit according to the procedure described above.

Some issues of Low and informational severity were found; some suggestions and best practices are also provided in order to improve the code quality and security posture.

In the end, BabyDoge Team Resolved all issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the BabyDogeCoin Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the BabyDogeCoin Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2023

For

BabyDoge

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com