QuillAudits

Audit Report
September, 2022

For

Table of Content

Executive Summary

Project Name	BoredLucky
Timeline	22th August 2020 to 28 August 2022
Method	Manual Review, Functional Testing, Automated Testing.
Scope of Audit	The scope of this audit was to analyse BoredLucky codebase for quality, security, and correctness. Git Repo link: *https://github.com/boredlucky/raffle-contracts* Git Branch: Main Commit Hash: 0ea75a7354657ecaca30b69b9c078a7551ba6a9d

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	0	0

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
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

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

A. Contract - Raffle.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

No issues found

Functional Tests

- purchaseTicket() must be in SellingTickets state
- purchaseTicket() reverts if purchase after endTimestamp
- purchaseTicket() reverts if purchase 0 tickets
- purchaseTicket() must have correct value of ETH
- giveawayTicket() must be in WaitingForStart or SellingTickets state
- giveawayTicket cannot giveaway after endTimestamp
- verifyNFTPresenceBeforeStart() can be called by anyone
- verifyNFTPresenceBeforeStart() can be called only in WaitingForNFT state
- cancelBeforeStart() can be called only in WaitingForNFT or WaitingForStart state
- cancelIfUnsold() can be called only in SellingTickets state
- cancelIfNoRNG() can be called only in WaitingForRNG state
- transferTicketRefundIfCancelled() can be called only in Cancelled state
- transferNFTToOwnerIfCancelled() can be called only in Cancelled state
- transferNFTToWinnerIfCompleted() can be called only in Cancelled state
- transferETHToOwnerIfCompleted() can be called only in Completed state
- getPurchasedTicketCount() should return the number of purchased tickets for given owner
- getState() should return the current state
- isWinnerDrawn() should return a correct _winnerDrawn
- getWinnerAddress() should return a correct _winnerAddress
- getWinnerDrawTimestamp should return a correct _winnerDrawTimestamp
- getWinnerTicketNumber should return a correct _winnerTicketNumber
- fulfillRandomWords() should have a correct `requestId`
- fulfillRandomWords() can be called only in WaitingForRNG state

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.

Slither

```
enderphan@enderphan contracts % slither TicketStorage.sol
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.8']
        - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.8 (TicketStorage.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
TicketStorage._assignTickets(address,uint16) (TicketStorage.sol#97-115) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.8 (TicketStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.8 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable TicketStorage._tickets (TicketStorage.sol#26) is not in mixedCase
Variable TicketStorage._ticketsLeft (TicketStorage.sol#27) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#61-63)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#69-72)
getTickets() should be declared external:
        - TicketStorage.getTickets() (TicketStorage.sol#44-46)
getTicketsLeft() should be declared external:
        - TicketStorage.getTicketsLeft() (TicketStorage.sol#51-53)
getTicketNumberRange(uint16) should be declared external:
        - TicketStorage.getTicketNumberRange(uint16) (TicketStorage.sol#58-60)
getAssignedTicketCount(address) should be declared external:
        - TicketStorage.getAssignedTicketCount(address) (TicketStorage.sol#65-67)
getAssignedTicketNumberRange(address,uint16) should be declared external:
        - TicketStorage.getAssignedTicketNumberRange(address,uint16) (TicketStorage.sol#76-77)
getAssignedTicketNumberRanges(address) should be declared external:
        - TicketStorage.getAssignedTicketNumberRanges(address) (TicketStorage.sol#85-87)
findOwnerOfTicketNumber(uint16) should be declared external:
        - TicketStorage.findOwnerOfTicketNumber(uint16) (TicketStorage.sol#125-150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solhint

```
Raffle.sol
   2:1   error    Compiler version ^0.8.8 does not satisfy the ^0.5.8 server requirement                          compiler-version
  84:5   warning  Explicitly mark visibility of state                                                            state-visibility
  84:5   warning  Variable name must be in mixedCase                                                             var-name-mixedcase
  85:5   warning  Explicitly mark visibility of state                                                            state-visibility
  86:5   warning  Explicitly mark visibility of state                                                            state-visibility
  90:5   warning  Explicitly mark visibility of state                                                            state-visibility
  91:5   warning  Explicitly mark visibility of state                                                            state-visibility
  92:5   warning  Explicitly mark visibility of state                                                            state-visibility
  94:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
 106:9   warning  Error message for require is too long                                                          reason-string
 106:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 107:9   warning  Error message for require is too long                                                          reason-string
 140:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 140:53  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 146:9   warning  Error message for require is too long                                                          reason-string
 146:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 147:9   warning  Error message for require is too long                                                          reason-string
 179:9   warning  Error message for require is too long                                                          reason-string
 185:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 185:53  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 189:9   warning  Error message for require is too long                                                          reason-string
 189:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 241:9   warning  Error message for require is too long                                                          reason-string
 259:9   warning  Error message for require is too long                                                          reason-string
 259:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 274:9   warning  Error message for require is too long                                                          reason-string
 274:17  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
 482:32  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time

✖ 28 problems (1 error, 27 warnings)
```

```
TicketStorage.sol
   2:1   error    Compiler version ^0.8.8 does not satisfy the ^0.5.8 server requirement                          compiler-version
  33:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  34:9   warning  Error message for require is too long                                                          reason-string
  35:9   warning  Error message for require is too long                                                          reason-string
  99:9   warning  Error message for require is too long                                                          reason-string
```

Closing Summary

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

No severity issues were found during the course of the audit

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the BoredLucky 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 BoredLucky 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.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com