QuillAudits

Audit Report
July, 2022

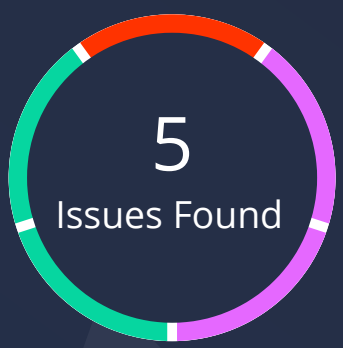
For

B4REAL

Table of Content

Executive Summary

Project Name	B4Real
Timeline	15 June, 2022 - 5 July, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse B4Real codebase for quality, security, and correctness. *https://polygonscan.com/address/0xaf294e4e2fe65c5de28bd0a3a073e6cf10d7ef6e#code*
Fixed In	*https://kovan.etherscan.io/address/0xA507Ee4410c8D5888c3bf29F5c178b07A62d9201*



High
Medium
Low
Informational

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

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

A. Contract - B4Real.sol

High Severity Issues

A.1 Transfer fee bypass

Function - transferFrom()

Description

If the "to" address is listed as true on the whitelist mapping and “waiveFees” is set to false, the fees payment can be bypassed by using the “approve” and “transferFrom” function. According to the intended behavior, if "to" address is listed as “true” on the whitelist mapping and “waiveFees” is set to “false”, the fees should "always" be deducted and there must not be any other way. However, if users use transferFrom instead of transfer, they can avoid paying any fees regardless of the whitelist.

Remediation

Make sure to add the fee deduction functionality in the “transferFrom” function.

Status

Resolved

Medium Severity Issues

No issues were found

Low Severity Issues

A.2 No maximum transfer fee

Function - setTaxFee()

Description

Admin can set transaction fees to whatever value they want (even 100%). It should be capped and must not be allowed exceed a certain limit

Remediation

We would recommend to set a maximum fee percentage variable.

Status

Partially Resolved

A.3 Missing events

Description

Missing events don't pose any security risk. However, it makes it difficult to track chain events. It is advisable to emit an event whenever a significant action takes place on contract.

Remediation

Consider adding events to the following functions:
- setTaxFee
- exemptFromFee
- includeInFee
- updateB4REALTaxAddress
- setAdmin
- transferOwnership

Status

Resolved

Informational Issues

A.4 Unlocked pragma (pragma solidity ^0.8.5)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Resolved

A.5 Transfer Ownership should be a two step process

Function - transferOwnership()

Description

The transfer of ownership is crucial functionality in token contracts and it should be done by a two step process so that the owner doesn't mistakenly transfer ownership to a false address.

Remediation

Use a two way process so that the owner can verify the address once again after passing it in the function.

Status

Acknowledged

General Recommendations for Gas Optimization

Description

In conclusion, we would like to mention that some of the public functions that are never called from within the contract should be declared external in order to save gas if there are no plans to inherit this contract in the future.

- setAdmin
- transferOwnership
- updateB4REALTaxAddress
- includeInFee
- exemptFromFee
- toggleTransactionFees
- setTaxFee
- updateB4REALTaxAddress

We would also like to conclude that there is functionality to transfer ownership of the contract and to set a new account as “Admin” as well but there is no feature to revoke the admin role directly by the B4REAL.sol contract.

Status

Resolved

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 B4Real. We performed our audit according to the procedure described above.

Some issues of High, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the End,Blacktie team Resolved almost all issues and Acknowledged one issue.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the B4Real 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 B4Real 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

B4REAL

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com