MixBytes()

ROOBEE

TOKEN SMART CONTRACTS SECURITY AUDIT REPORT

APRIL 18

2019

FOREWORD
TO REPORT

A small bug can cost you millions. **MixBytes** is a team of experienced blockchain engineers that reviews your codebase and helps you avoid potential heavy losses. More than 10 years of expertise in information security and high-load services and 11 000+ lines of audited code speak for themselves.

This document outlines our methodology, scope of work, and results.

We would like to thank **Roobee** for their trust and opportunity to audit their smart contracts.

CONTENT
DISCLAIMER

This report was made public upon consent of **Roobee**. **MixBytes** is not to be held responsible for any damage arising from or connected with the report.

Smart contract security audit does not guarantee a comprehensive inclusive analysis disclosing all possible errors and vulnerabilities but covers the majority of issues that represent threat to smart contract operation, have been overlooked or should be fixed.

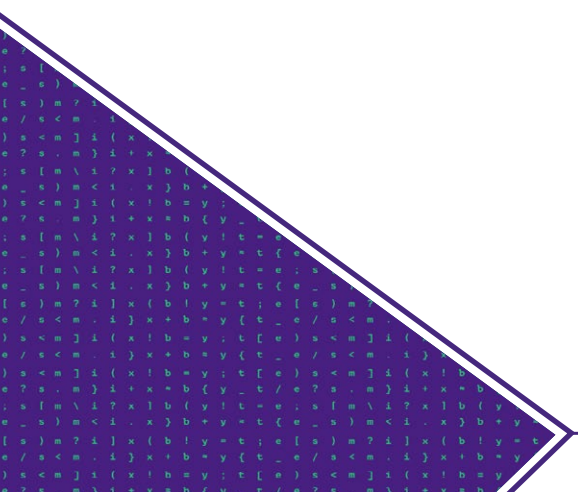


TABLE OF CONTENTS

01 INTRODUCTION TO THE AUDIT

GENERAL PROVISIONS

The **Roobee** team asked **MixBytes** to audit their token and vesting smart contracts. The code was located in the following **github repository**.

SCOPE OF THE AUDIT

AUDITED OBJECT	LOCATION
Smart contracts	IERC20.sol
	Migrations.sol
	Roles.sol
	Roobee.sol
	SafeMath.sol

Audited commit: 12ff7eb423aedf426c8e1389a686ace0eda2eddb.

02 SECURITY ASSESSMENT PRINCIPLES

CLASSIFICATION OF ISSUES

CRITICAL

Bugs leading to Ether or token theft, fund access locking or any other loss of Ether/tokens to be transferred to any party (for example, dividends).

MAJOR

Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.

WARNINGS

Bugs that can break the intended contract logic or expose it to DoS attacks.

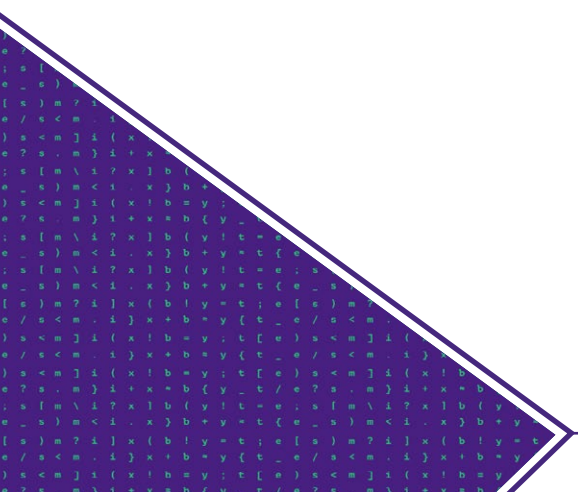
COMMENTS

Other issues and recommendations reported to/acknowledged by the team.

SECURITY ASSESMENT METHODOLOGY

The audit was performed with triple redundancy by three auditors. Stages of the audit were as follows:

1. “Blind” manual check of the code and model behind the code
2. “Guided” manual check of the code
3. Check of adherence of the code to requirements of the client
4. Automated security analysis using internal solidity security checker
5. Automated security analysis using public analysers
6. Manual by-checklist inspection of the system
7. Discussion and merge of independent audit results
8. Report execution



03 DETECTED ISSUES

MAJOR

None found.

CRITICAL

None found.

WARNING

1. Roobee.sol:336

Here and further on: there is a minimal risk that the compiler optimizer will reject the entire string despite the presence of the side effect (revert), since the result is not used. The compiler is still young and under active development as evidenced by major version zero.

We do not recommend using this construction. Instead, it is better to explicitly write `require (getAvailableBalance (msg.sender)> = value)`.

Status:

FIXED - in **16431b1**

2. Roobee.sol:371

After `approvalFallback` execution, the allowance that the `_spender` had before calling the `approveAndCall` function will not be returned to its original value. Potentially, this enables `_spender` not to spend tokens immediately and not to perform the operation requested in `_extraData`, but to write off the tokens later, using the remaining allowance.

We recommend returning the allowance for `_spender` to its original value before ending the `approveAndCall` function.

Status:

ACKNOWLEDGED

3. Roobee.sol:268-269

Token `name` and `symbol` differ from those stated in the **documentation** by the case of characters.

Status:

FIXED - in **5eb2b41**

4. Roobee.sol:316

We recommend adding a check that the `_unfreezeTimestamp` value does not exceed reasonable limits.

Status:

ACKNOWLEDGED

COMMENT

1. Roobee.sol:294

In case the user executes `_subsequentUnlock`, then when the time reaches `_unfreezeTimestamp`, all user tokens will still remain frozen for 30 days.

2. Roobee.sol:357

The previous allowance value of `_spender` will be reset at this point.

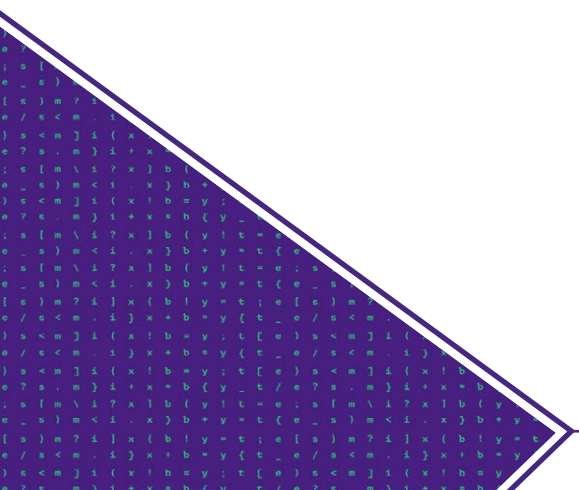
3. Roobee.sol:280

We recommend using `decimals` instead of duplicating `1e18`.

04 CONCLUSION AND RESULTS

We haven't identified any high risk vulnerabilities in the audited smart contracts. Overall code quality is on a high level. We suggested some corrections and code optimization solutions.

As of commit **16431b1**, no issues were identified in the smart contract code.



ABOUT MIXBYTES

MixBytes is a team of experienced developers providing top-notch blockchain solutions, smart contract security audits and tech advisory.

JOIN US

OUR CONTACTS

Alex Makeev
Chief Technical Officer

Vadim Buyanov
Project Manager