MixBytes()

Akropolis

TOKEN SALE CONTRACTS AUDIT REPORT

JUNE 30

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 **Akropolis** for their trust and opportunity to audit their smart contracts.

CONTENT
DISCLAIMER

This report was made public upon consent of **Akropolis**. **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 **Akropolis** team asked **MixBytes Blockchain Labs** to audit their token sale contracts. The code was located in the hidden github repository.

SCOPE OF THE AUDIT

The primary scope of the audit is smart contracts located at: **https://github.com/akropolisio/AkropolisToken/tree/3ad8eaa6f2849dceb125c8c614d5d61e90d465a2/contracts**.

The scope is limited to contracts which are used in migrations at: **https://github.com/akropolisio/AkropolisToken/tree/3ad8eaa6f2849dceb125c8c614d5d61e90d465a2/migrations**.

Audited commit is **3ad8eaa6f2849dceb125c8c614d5d61e90d465a2**.

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

CRITICAL

None found.

MAJOR

1. Collision of storage layouts of TokenProxy and AkropolisToken

The problem is illustrated by the `test/TestProxySlotCollision.js` (works for commit **3ad8eaa6f2849dceb125c8c614d5d61e90d465a2**).

As can be shown, a collision is almost completely avoided because `paused` and `locked` flags were packed by the solidity compiler and don't collide with other fields, as well as the slot for whitelist not being used (because mappings are implemented in such way). But there is collision of `bool whitelisted` and `decimals` fields.

A simple solution is to use "unique" slot locations for each field (except shared base contract fields) derived via `keccak256`, for example: **https://github.com/poanetwork/poa-network-consensus-contracts/blob/0c175cb98dac52201342f4e5e617f89a184dd467/contracts/KeysManager.sol#L185**.

In this case we also recommend that the contract name into hash function invocation is included, and the use of `abi.encode` in place of `abi.encodePacked`, like this: `uintStorage[keccak256(abi.encode("TokenProxy", "decimals"))] = decimals`.

Status:

FIXED - in commit **79565a3**

WARNINGS

1. Lockable.sol#L25

A variable is named inversely to its value, meaning “unlocked” is to be expected in this case. Normally variable names are not a critical issue, but in this case as a result of code modifications during maintenance, it may lead to logic reversal.

Status:

FIXED - in commit **28a4153**

2. AkropolisToken.sol#L41

The result of a function call from the base contract is ignored and the result is always returned as `false`. Any users of the “AkropolisToken” contract (including other smart-contracts) who check the result of the function, will consider calls to have failed. Most likely, the following piece of code is missing `return super.approve(...)`.

Status:

FIXED - in commit **7dee846**

3. AkropolisToken.sol#L75

The result of a function call from the base contract is ignored and the result is always returned as `false`. Any users of the “AkropolisToken” contract (including other smart-contracts) who check the result of the function will consider calls to have failed. Most likely, the following piece of code is missing `return super.transfer(...)`.

Status:

FIXED - in commit **7dee846**

4. AkropolisToken.sol#L92

The result of a function call from the base contract is ignored and the result is always returned as `false`. Any users of the “AkropolisToken” contract (including other smart-contracts) who check the result of the function, will consider calls to have failed. It appears that the following piece of code is missing `return super.transferFrom(...)`.

Status:

FIXED - in commit **7dee846**

5. AkropolisToken.sol#L11

The `approve` function is not disabled by default, contrary to what the comment claims. Moreover, there is a contradiction with **this commentary** - the `approve` function is not blocked by a designated mechanism or a flag. It’s allowed by the common pause mechanism, also implemented for the following functions: `increaseApproval`, `decreaseApproval`, `transfer`, `transferFrom`. Modifier `whenUnlocked` is removed in the following commit **434aab**.

Status:

FIXED - in commit **28a4153**

COMMENTS

1. DelayedUpgradeabilityProxy.sol#L17

We recommend declaring `UPGRADE_DELAY` as `constant`. This will prevent unintended modifications and save gas.

Status:

ACKNOWLEDGED

2. Solidity 0.5

We recommend updating the compiler to version 0.5 or newer, as it includes error fixes and a lot of smaller tweaks and checks, facilitating safe code writing.

Status:

ACKNOWLEDGED

04 CONCLUSION AND RESULTS

The use of proxy-contracts mechanism in Solidity and EVM has its risks. We detected and suggested a fix to a problem that arose in connection with it. A number of minor issues were also addressed. The rest of the code is well-structured and written perfectly.

The version **2e353cf** doesn't have any vulnerabilities or weak spots according to the analysis.

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