Source Code Review

Prepared for Annihilat.io• December 2017

V1.0

1. Table of Contents

2. Executive Summary

In December 2017, Annihilat.io engaged Coinspect to perform a security audit of the Annihilat.io token contracts. The contract was received in the following repository branch:

https://github.com/annihilatio/ido commit 763854b6256a7f1c0c8d1d494f0d832a1b09ddaf

The objective of the audit was to evaluate the security of the smart contracts. During the assessment, Coinspect identified the following issues:

High-Risk	Medium Risk	Low Risk	No Risk
0	3	1	1

3. Introduction

The contract Annihilat.io token comprises the following contracts:

- Token_flat.sol
- Multisig_flat.sol

Token_flat.sol is an ERC-20 compliant token smart contract, that also provides the functionality to manage token generation events. The Token_flat is the main contract that buyers and users will interact with. The requirements and basic functionality of these contracts is described in the document SMART-CONTRACT-SPECS.md in the github repository.

A whitebox security audit was conducted on these smart contracts.
The present report was completed on December 11th, 2017, by Coinspect. The report includes all issues identified in the audit.

The following checks, related to best practices, were performed:

- Confusion of the different method calling possibilities: send(), transfer(), and call.value()
- Missing error handling of external calls
- Erroneous control flow assumptions after external calls
- The use of push over pull for external calls
- Lack of enforcement of invariants with assert() and require()
- Rounding errors in integer division
- Fallback functions with higher gas limit than 2300
- Functions and state variables without explicitly visibility

- Missing pragmas to for compiler version
- Race conditions, such as contract Reentrancy
- Transaction front running
- Timestamp dependence
- Integer overflow and underflow
- Code blocks that consumes a non-constant amount of gas, that grows over block gas limit.
- Denial of Service attacks
- Suspicious code or underhanded code.
- Error prone code constructs
- Race conditions

4. Findings

The list of issues found follows.

4.1. Unnecessary call to transfer() leads to more gas consumption

No Risk

In the Token_flat fallback function the refund is made even if the refund amount is zero. This adds 1000 extra gas cost to the transaction.

Recommendation

Only execute msg.sender.transfer(refundAmount) if refundAmount is non-zero.

4.2. Replaceable IToken in Multisig_flat

Medium Risk

Any of the owners of the Multisig_flat wallet can change the token associated with the multi-signature wallet at any time. This implies that any owner can prevent the execution of tgeSettingsChangeRequest() and confirmGoLive() and cancelGoLive().
If the change of token is done just before the execution of any of these methods and restored just after, it's possible that the change of token to go unnoticed by the remaining owners, who will believe that the method has been executed on the Anihilatio token, when if fact it has not.

Recommendation

Add the "onlyWallet" modifier to setToken() or prevent the change of the token if already set.

4.3. confirmSettingsChange can be replayed

Medium Risk

The operations that change TGE settings can be replayed. One of the owners of the multi-sig can force a change of settings by restoring a previously used setting.

Recommendation

Require that request.executed = false as a condition to execute confirmSettingsChange()

4.4. confirmGoLive() and cancelGoLive() can be replayed

Medium Risk

After the first call to confirmGoLive() or cancelGoLive(), these commands are replayable by any of the owners, without the need of approval by the other requested owner signatures.

Recommendation

- Instead of using independant quorum functions based on frozenConfirms, use submitTransaction() where the destination is the Multisig_flat itself and the data contains the 4-byte method signature of confirmGoLive() or cancelGoLive().
- Remove the ownerExists(msg.sender) modifier from confirmGoLive() and cancelGoLive().
- Add the modifier onlyWallet() to confirmGoLive() and cancelGoLive().

4.5. No automatic TGE

Low Risk

The documentation states that “TGE automatically goes Live when there is less than 1 Token left in project multisig wallet.“. This functionality is not implemented.

Recommendation

Implement the automatic TGE functionality or remove the requirement.

5.Closing Remarks

The scope of the present security audit is limited to smart contract code. It does not cover the technologies and designs related to these smart contracts, nor the frameworks and wallets that communicate with the contracts, nor the operational security of the company that created and will deploy the audited contracts. This document should not be read as investment or legal advice.