FORWARDER
Smart Contracts

Mikhail Vladimirov and Dmitry Khovratovich

06th September 2018

This document describes the audit process of the Forwarder smart contract performed by ABDK Consulting.

1. Introduction

We've been asked to review the Forwarder smart contract given in private access to the Forwarder repository.

2. Forwarder

In this section we describe issues related to the smart contract defined in the Forwarder.sol.

2.1 EIP-20 Compliance Issues

This section lists issues of token smart contract related to the EIP-20 requirements.

1. Line 10: the contract `ERC20` is not ERC20 nor EIP-20 compliant despite the name. It does not emit events required by the standard and does not implement methods `approve`, `allowance`, and `transferFrom`.
2. Line 17: according to EIP-20: *a token contract which creates new tokens should trigger a Transfer event with the* `_from` *address set to* `0x0` *when tokens are created*.
   This will help blockchain explorers to recognize contract creator as token holder immediately after contract deployment.

2.2 Readability Issues

This section lists cases where the code is correct, but too involved and/or complicated to verify or analyze.

Line 14: `10e18` instead of `10 * (10 ** 18)` would be more readable.

2.3 Suboptimal Code

This section lists suboptimal code patterns, which were found in the smart contract.

1. Line 12,14: the access is not specified, in this case the `internal` would be used as default.
2. Line 14: Solidity code style does not use underscore at the end of the function.
3. Line 69: in this line a condition `msg.value>0` might be useful.
4. Line 73,121: typecasts `address(this)` and `address(pf)` are redundant. All smart contract types inherit from address.
5. Line 78: the return value may be declared as bool result making separate local variable unnecessary.
6. Line 82: the assembly section looks equivalent to `result = destination.call.gas(gasleft()-34710).value(value)(data);` Solidity statement.
7. Line 108: perhaps, there is no need to store the contract addresses. The array is not accessed by any contract method, whereas backends can just scan events.

2.4 Unclear Behaviour

This section lists issues of the token smart contract, where the contract behavior is unclear: the business logic might be violated here, but the documentation and functional requirements are not sufficiently documented to make a clear decision.

Line 108: the statement will generate a public getter names `recipients` with two parameters: recipient address and contract index. Is this desired?

2.5 Moderate Issues

This section lists major issues which were found in the token smart contract.

Line 72: the method `sweep` can be called by everyone. So the malicious actor may prevent call to `externalCall` with non-zero value from being successfully executed, by front running it with `sweep` call. Consider protection `sweep` somehow.

2.6 Other Issues

This section lists stylistic and other minor issues which were found in the token smart contract.

1. Line 16,78: names of parameters are without underscore.
2. Line 32: the method `transfer` should emit `Transfer` event.
3. Line 62: perhaps, there is no need to index the value.

3 Our Recommendations

Based on our findings, we recommend the following:

1. Check the moderate issue.
2. Make the token contract EIP-20 compliant if it is supposed to be so.
3. Check issue marked “unclear behavior” against functional requirements.
4. Refactor the code to remove suboptimal parts.
5. Fix the readability and other (minor) issues.