CYBER3

EXPERT REPORT

on the results of security audit
of the smart contract
of the AA.M Group company

Table of Contents

Introduction

General provisions

This expert report contains the results of security audit of the source code (hereinafter referred to as “the System”) owned by AA.M Group (hereinafter referred to as “the Company”) with recommendations on the current security level improvement.

Generally accepted abbreviations

Abbreviation	Meaning
IS	Information security
DS	Data system
EVM	Ethereum virtual machine
ERC20	Adopted Token standard in the Ethereum ecosystem
Ether	Ethereum network cryptocurrency
ICO	Initial coin offering

Table 1. Generally accepted abbreviations

Abstract

Cyber3’s experts conducted the security audit of the System. In the scope of work,

they used and examined the following attacker models:

- External attacker from the Ethereum network
- Internal attacker (an owner or system operator)

The performed works indicated that:

1. The current security level of the System could be rated “average”
2. One critical vulnerability was discovered

The list of key recommendations is as follows:

1. Fix discovered vulnerabilities and security weaknesses

Below, you can find the detailed description of the discovered security weaknesses and the risks associated with them as well as the list of recommendations on how to fix them.

Security Audit principles

IS threats

There are 3 types of IS threats that can affect a Company's information resources: violations of confidentiality, integrity or availability of information.

Confidentiality violations are usually aimed at information disclosure. If a violation is successful, the information becomes known to people, who should not have access to it: unauthorized personnel of the Company, clients, partners, competitors, and third parties.

Integrity violations are aimed at modification or corruption of the information, which can lead to modification of its structure or content, and to complete or partial destruction of the data.

An availability violation (denial-of-service threat) involves data system users' inability to access the information.

The core principle of this IS audit is an estimation of exploitation likelihood of the aforementioned threats affecting the Company's information resources, within the framework of a pre-defined attacker model.

Attacker model

A potential attacker is an individual or a group of individuals, acting either in a collusion or independently, whose intended or unintended actions can carry the aforementioned threats to the IS, infringe information resources of the System or negatively affect the Company's interests.

IS threats are basic threats to confidentiality and integrity of information and the threat of the System denial-of-service.

Attackers can pursue the following goals (and their possible combinations):

- Cause Denial of Service
- Escalate their privileges in the System
- Get unauthorized access to business-critical data

In the scope of work, Cyber3's experts used external and internal attacker models.

External attacker

During the security audit, the experts used the following submodel of an external attacker:

- External Internet attacker who has access to the Ethereum network and knowledge about the tested System (because the source code of the smart contract is open), but no privileges in the network

Internal attacker

In order to conduct penetration testing, the experts used the following submodel of an internal attacker:

- Internal attacker from the Ethereum network who has both the knowledge about the tested System and privileges in it

Scope of works

During the audit, the following code was used: rinkeby.etherscan.io

Smart contract	Description
SafeMath	Safe math operations
Ownable	Segregation and transfer of ownership
Pausable	Function blocking by an owner
ERC20	ERC20 – abstract class
StandartToken	Functionality of the ERC20 token
BurnableToken	"Token burning" functionality
MintableToken	Functionality of an "issued" token
Manageable	"Add/Remove" manager functionality
Token	Properties of the ALL.ME token
Crowdsale	Sale contract of the ALL.ME token

ICO scenario

Before the ICO is initiated, there are 3 bln. ME tokens for internal use (user deposits, rewards, etc.). After that, the price of a token for the first wave of sales is set, and the sales are started.

When a particular amount of tokens is sold, an owner stops the sales with the pause() function. To initiate the second wave, an owner calls the unpause() function. The algorithms will be followed up to the last wave of sales.

During the ICO, the price of the ETH may vary. It makes it possible for an owner to pause sales at any moment and adjust the pricing.

Other features:

- Token issuing is limited (there are only 10 bln. tokens, which are issued during Crowdsale)
- Initial pricing: 1 ETH = 200 tokens
- Minimal purchase sum: 0.001 ETH
- Tokens for sale: 7 bln

- The sum gathered from token purchases is sent to a beneficiary
- Crowdsale is finished with the withdraw() function: control over a token is passed to a beneficiary
- Token price is changed with the setTokenPrice(_value) function, where _value – is a number of tokens purchasable for 1 Ethe0r. vThe price can be changed by an administrator only when the sales is paused. After Crowdsale is finished, the function becomes unavailable.

List of discovered vulnerabilities

Round-up error during token purchase

Audit of smart contract source code

Severity: high

Likelihood of exploitation: high

Overall impact level: high

Description:

While the amount of purchased tokens is calculated, the fact that EVM drops fractions while dividing types is neglected.

Impact:

Due to the dropped fractions of purchased tokens, a user loses some tokens. The lost amount of tokens may make a considerable sum for a user and, consequently, affect the Company's reputation.

Vulnerable resource:

- Crowdsale

Technical details:

To calculate the number of tokens that will be entered to a user's account, the following code is used:

```
uint public priceTokenWei = 1 ether / 200;
// 5000000000000000
uint sum = msg.value;
// 333333333333333333
uint amount = sum.div(priceTokenWei).mul(1 ether);
// 66000000000000000000
```

Fractions are dropped at the sum.div(priceTokenWei) stage.

If the purchase() function with 0.333333333333333333 ETH is called, the amount will be equal to 66 tokens instead

of 66.666666666666666600. In other words, 333333333333333333 / 5000000000000000 = 66.

Recommendations:

- Use the RATE pattern to calculate the amount
- Or change the order of multiplications and divisions during amount calculation

List of discovered security weaknesses

Redundant functionality

Description:

The burn function in the BurnableToken function implements the "token burning" functionality for its caller.

Vulnerable resource:

- BurnableToken

Recommendations:

- Since the crowdsale scenario and the use of tokens do not imply token burning, it is recommended to get rid of it

Token transfers are available when sales are paused

Description:

According to the ICO scenario, token sales are paused to split the process into several stages. Nevertheless, the System does not restrict token transfers among the participants.

Impact:

By using the pauses, some participants may sell tokens at the prices they have set themselves.

Vulnerable resource:

- StandardToken

Technical details:

In the System, the sales pause functionality is implemented with the help of the whenNotPaused and whenPaused modifiers of the Pausable contract. However, they are applied for the functions of the Crowdsale contract.

Recommendations:

- Restrict availability of the transfer and transferFrom functions while sales are paused

Non-fixed compiler version

Description:

During the final deployment, a contract is compiled by the same version of the compiler it has been developed and tested with.

Impact:

Fixing compiler version in the source code may help avoid the deployment of a smart contract compiled with a newer compiler version, which is more likely to have unknown issues.

Vulnerable resource:

- Final smart contract

Technical details:

The pragma solidity implements the selection of a compiler version.

Recommendations:

- Set a compiler version. For example, pragma solidity 0.4.18; (note, there is no ^)

The Manager cannot close the ICO

Description:

The System allows token purchases in other currencies with the help of a manager.

Impact:

Due to the strict check for a maximum token amount being exceeded, a manager will not be able to close ICO.

Vulnerable resource:

- Crowdsale

Technical details:

In the externalPurchase function, there is a strict check for a maximum token amount being exceeded. Due to this, a manager will not be able to close sales (with a final purchase).

```
require(tokensSold.add(_value) < tokensForSale);
```

Recommendations:

- Change < to <=

Style notes

Comments

Description:

The Token follows a comment that reflects the specification in Russian. However, the comments for the Crowdsale contract are written in English.

Recommendations:

- Write all comments in English

Function name does not correspond its purpose

Description:

The withdraw function of the Crowdsale contract implements the functionality of closing the ICO and changing Token ownership. However, such a name is usually used for functions responsible for withdrawal.

Recommendations:

- Use a proper name that corresponds to the purpose of the function

4

License

Appendix 1. Security audit

Security level analysis

To analyze the System security level, it is necessary to measure severity and likelihood of the detected vulnerabilities exploitation. The likelihood of exploitation is measured, according to the ease of vulnerability exploitation and the accessibility of a vulnerability.

Vulnerability analysis

The “Severity” property of a vulnerability describes possible results of this vulnerability exploitation, regarding confidentiality, integrity, and availability of information processed on a vulnerable resource. Severity levels are described in the Table A-1.

Severity level	Confidentiality violation	Integrity violation	Availability violation
None	Does not happen	Does not happen	Does not happen
Low	Obtaining access to a noncritical information by an attacker through privilege escalation	Integrity violation of a noncritical information by an attacker with basic user rights in the System	Short-time denial-of-service of a mission-critical application
Medium	Confidentiality violation of sensitive data by an attacker with basic user rights in the System	Integrity violation of sensitive data by an attacker with basic user rights in the System	Denial of service of a mission-critical application or a short-time denial of service of the System
High	Confidentiality violation of critical information by an attacker with administrator rights in the System	Integrity violation of critical information by an attacker with administrator rights in the System	Denial of Service of the System

Table A–1. Severity levels

Ease of vulnerability exploitation

The "Ease of exploitation" property of a vulnerability defines what hardware and software, time and computing resources, and professional skills are required to exploit a vulnerability (Table A-2).

Level	Description
Low	Vulnerability exploitation requires high computing powers, significant time resources, developing new software, configuration analysis of the System, determination and testing possible ways and conditions of successful exploitation of this vulnerability.
Average	Vulnerability exploitation requires high-performance computing, extensive time resources, special hardware and software, and analysis of a violated system configuration. An attacker does not have to have deep knowledge of the system or professional skills to perform an attack.
High	Vulnerability exploitation does not require the use of any special hardware or software, high-performance computing, time resources or any professional skills to perform an attack.

Table A–2. Ease of vulnerability exploitation levels

Vulnerability availability

The "Availability" property of a vulnerability defines what user classes have access to a vulnerable resource (Table A-3).

Level	Description
Low	Privileged users
Average	Registered users
High	All users

Table A–3. Availability levels

Likelihood of exploitation

The likelihood of exploitation is calculated according to “ease of exploitation” and “accessibility” levels (Table A–4).

Likelihood of exploitation		Ease of exploitation ↓		
		Low	Average	High
Availability →	Low	Low	Low	Average
	Average	Low	Average	High
	High	Average	High	High

Table A–4. Likelihood of exploitation levels

Vulnerability impact

Vulnerability impact (for one of the existing threats) is measured, according to vulnerability severity (for one of the existing threats) and the likelihood of exploitation of a vulnerability (Table A-5).

Vulnerability impact		Likelihood of exploitation ↓		
		Low	Average	High
Vulnerability severity →	Low	Low	Low	Average
	Average	Low	Average	High
	High	Average	High	High

Table A–5. Vulnerability impact levels

Appendix 2. The list of vulnerabilities and security weaknesses discovered in the System

The vulnerabilities and security weaknesses detected during this security audit are listed in Table B-1.

Discovered vulnerabilities		
Vulnerability	Impact level	See paragraph
Round-up error during the token purchase	High	3.1.1.

Discovered security weaknesses	
Weaknesses	See paragraph
Redundant functionality	3.2.1.
Token transfers are available when sales are paused	3.2.2.
Non-fixed compiler version	3.2.3.
The manager cannot close the ICO	3.2.4.

Table B-1. Vulnerabilities and security weaknesses discovered in the System

CYBER3