A Model for Estimating the Security Level of Mobile Applications: a Fuzzy Logic Approach

Olha Yanholenko [1][0000-0001-7755-1255], Olga Cherednichenko [1][0000-0002-9391-5220], Olena Yakovleva [2][0000-0002-6129-6146], Denis Arkatov [1][0000-0003-0162-159X]

[1] National Technical University “Kharkiv Polytechnic Institute”, Kharkiv, Ukraine
[2] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
olga.yan26@gmail.com, olha.cherednichenko@gmail.com, olena.yakovleva@nure.ua, denarkatov@gmail.com

Abstract. In this paper a model for solving the problem of estimating the security level of mobile applications was proposed. The estimation is performed based on a fuzzy inference system of the Mamdani type. The input criteria were defined as the most important security threats by applying the Analytic hierarchy process method. The pairwise comparison matrix was constructed from mobile security research on OWASP Top 10 Mobile Risks. The proposed methodology can be applied for any kind of mobile applications available for modern platforms, except specific cases when security analyst does not have a sufficient amount of information about the chosen application for performing the security level testing. Mobile security analysts can easily make further decisions about comprehensive mobile application security based on the results obtained with the help of the introduced model.

Keywords: Mobile Application; Security; Model; Fuzzy Logic; Analytic Hierarchy Process; Comprehensive Mobile Application Security.

1 Introduction

The number of available applications for mobile platforms reached 2 million on Apple AppStore and over 2.2 million available on Google Play Store [1]. An exponential growth of portable application development and a large number of existing vulnerabilities made them enthralling for a wide variety of potential attackers. According to the study published by [2], a total number of mobile malware reached 16 million by Q2 2017. Pradeo’s Tech team has shown in their recently published Mobile Applications Threats Review that nearly 60% of modern mobile applications contain security vulnerabilities [3]. Moreover, 1 out of 4 apps has security risks listed in Top 10 Mobile Risks of the OWASP Foundation [4]. Gartner analysts say that nearly 75% of apps failed security vulnerabilities checks [5]. Due to the relatively new market, there is a visible lack of approaches to testing the apps security available for mobile devices. Recently published research papers mainly focus on specific aspects of safety testing (for example, communication protocol testing [6]; GUI testing [7]; testing methodology based on privacy information encoding mechanism [8]; malware detection in mobile devices [9]; and software model checking approach [10] which does not reveal entire picture of potential app threats.

Taking into account the mobile threats growth during the last few years and a visible lack of comprehensive testing techniques it is possible to state that mobile application security evaluation continues to be an urgent problem. Therefore, the goal of the given research is to analyze modern security methodologies and develop a new approach which would help security analysts to estimate the overall mobile application security level. The goal is to be reached in three major steps:

1) Firstly, defining most important vulnerabilities, which presence significantly impacts the overall application security. By applying the Analytic Hierarchy Process (AHP) to matrix constructed from expert analysis data published by OWASP [4] in Top 10 Mobile Risks list of the most valuable threats will be constructed.

2) Secondly, constructing a fuzzy logic model for estimation of the security level of mobile application.

3) Finally, creating software testing tool which implements aforementioned algorithm and analysis of the received results.

The rest of paper is organized as follows. Section “State of the Art” introduces the background literature. It refers to the current scientific advances in mobile application security and modern techniques and challenges concerned with assessment, arrangement, and maintaining sufficient security of mobile applications, essential to avoid major intruding attacks. Based on performed literature analysis it is possible to define the main task of this work. Section “Methods” introduces main methods used for the construction of the proposed model. The main instruments described here are the Analytic Hierarchy Process, which uses the mobile security threats proposed by [4] as the inputs, and a fuzzy inference system (FIS) (in particular, of Mamdani type). In addition, this part provides a method for gathering input data, necessary for performing further app analysis. Section “Results” represents a step-by-step explanation of the case study organized in logical order. The description provided here can be used as a reference sufficient to replicate the proposed model. Section contains outcomes, obtained from real-life examples security testing. This part represents the security-testing results for eight open-source mobile applications along with all expert analysis, necessary for replicating the experiment manually. Section “Discussion” consists of a critical analysis of achieved improvements in comparison with results in related works. In addition, the limitations and weaknesses of the proposed solution are considered. Section “Conclusions and future work” summarizes the results and includes further ways of methodology improvement.

2 State of the Art

At the present, the well-known IT-companies such as Hewlett Packard, Gartner, OWASP, McAfee and Pradeo publish numerous reports on mobile application security. Experts working in this area focus their attention on the problem of various mobile applications security issues. Hewlett Packard Enterprise report [11] places the emphasis on the privacy issue. According to them, 97% of analyzed apps have access to at least one private data source, and such massive data collection could potentially go awry. The research team based their conclusions on a binary analysis of over 36,000 unique iOS and Android applications. McAfee's Strategic Intelligence researchers provided insights about modern mobile malware [2]. According to this study, there are three broad classes of evasion techniques: anti-security, anti-sandbox, and anti-analyst malware techniques. Total mobile malware amount continues to grow and reached over 16 million by Q1 2017. It is most widespread in Asia-region (more than 20% customers reported mobile malware). Mobile Application Security Study made by Hewlett Packard Enterprise [12] demonstrated in their report findings that nearly 75% of mobile applications do not utilize proper encryption techniques which could lead to data leakage. The conclusion was based on the scanning analysis of over 2,000 applications from 600 different corporate companies listed in Forbes list of Global 2000. The 2017 Mobile App Market predictions and analysis highlighted the fact that security issues are the center of attraction for the mobile developers nowadays [1]. The number of mobile security attacks will continue to grow tremendously. Gartner research team provided a constructive guide to prioritize mobile defenses [5]. According to this paper, since the sophisticated attacks become more widespread, companies must implement Mobile Threat Defense (MTD) techniques to avoid potential data and financial loses. Therefore, the leading actors and developers of the mobile apps market put a topic of their products' security at the top of their priorities.

At the same time, business and academic community work together in order to develop automatic tools for mobile apps security testing. The research by [6] described commonly used encryption protocol for secure communication between a mobile application (client-side) and a server-side. Summarizing their insights, it is possible to say that the use of outdated or compromised protocols for communication can cause serious leakage of private data. In addition, they reported a fundamental flaw in the improper usage of popular encryption TLS/SSL protocols, which could lead to the third-party data stealing.

The team of researchers introduced a tool "Autoforge", that can automatically send requests from the client app to check whether the server side handles user data with proper security measures [13]. It analyses server responses by imitating requests from the client side. They performed an experiment on 76 most popular applications (more than million installations) and found that 65% of apps back-end servers have login credentials brute-force vulnerability. Another software tool (iMPAcT) performs automatic testing based on analysis of UI Patterns, which is suitable for Android apps with graphical user interface (GUI) [7]. The software tool performs analysis by presence identification process from the predefined catalogue of all known patterns. Paper [14] shows different types of tests suitable for mobile applications. The paper considered five broad types of testing: functional, performance, usability, installation, and operational testing.

The research conducted by [8] proposed a method to analyze the functionalities related to privacy of client's information. The testing process is described using UI Automator Viewer, which is a part of Android SDK Tools. A comprehensive study by [9] demonstrated malware detection techniques based on sequence alignment algorithms. By applying proposed methods, it is possible to successfully identify and neutralize malware before it performs attacks. Authors of [15] recommend 40 possible ways to perform penetration testing; it describes software products for mobile security testing such as Burp Suite, Wireshark, MITM proxy and Androguard. The book focuses on iOS and Android platforms, but introduces basic testing methods for BlackBerry and Windows mobile platforms.

Another team of researchers announced a tool (K-Android), which is based on the K framework to test the effect of app-collision [10]. This condition can happen when several applications exchange data whereas they have an access to private information that could not be obtained per se. K-Android performs checking of APK-files, based on static security analysis methods. Netcraft’s research [16] on mobile security testing shows the required steps of comprehensive security threats analysis and the potential list of vulnerabilities. They focus attention on applications that use SSL encryption but have failed mechanisms to validate SSL-certificates, thus making them a potential target for the man-in-the-middle (MITM) attacks. In addition, they advertise The Mobile App Security Testing service performs security testing according to PCI DSS v2.0 requirement 11.3. The book [17] provides a list of various tools to perform penetration testing for Apple iOS developers. He describes iOS built-in security model, device Jailbreaking and TOP 10 Mobile security shortcomings identification and prevention methods. Most tests described in this book focus on static and dynamic analysis, penetration testing, and successful threat assessment.

OWASP Mobile Security Testing Guide proposed three major steps in security analysis, which are: Intelligence Gathering (security analyst gather as much information about application as possible to conduct further research), Threat Modeling (identifying threats by using published security reports and proposing countermeasures to prevent these threats) and Vulnerability Analysis (threats identification using test-cases) [18]. Vulnerability Analysis includes three broad classes of security testing methods: static methods (source code analysis), dynamic methods (network communication monitoring, analyzing running application) and forensic methods (app artifacts analysis, such as databases, log files, cookies, etc.).

The aforementioned tools have been developed based on the methods and algorithms of various theoretical frameworks. The methods applied can be classified as experience-based, simulation-based, and methods of mathematical modelling. Thus, the experience-based approach implies the execution of multiple tests on the app functioning and analysis of the obtained results. For instance, [6] measure the app security in terms of successfully passed tests of safe client-server communication guaranteed by TSL encryption. The techniques suggested by [15] are oriented on the collection of real testing results and making straightforward conclusions on their basis. The GUI testing tool developed by [7] is based on the pattern matching models represented via finite state machines and the logical rules of transition between states.

The simulation-based approach presupposes the intended injection of the malware messages/code/app into the mobile device. Then the estimation of the mobile app is performed with respect to its security under the created conditions. For example, the developers of the “Autoforge” software made it automatically generate a new input message with mutated fields satisfying the cryptographic constraints in a black-box manner [13]. The authors have analyzed 3 measures of message field differences (pattern matching, content matching, and the degree of difference – the Levenshtein distance calculated by the Wagner-Fischer algorithm). The difference occurs between a traced and forge messages when that last is simulated by the "Autoforge" in order to estimate the app security. Mathematical modelling methods provide the complex processing of security-related data collected during experiments with the targeted mobile app. Vidal [9] use the non-parametric Wilcoxon signed-rank test that operates on paired data vectors of the scores resulted from aligning the monitored sequences of system calls with the legitimate sample. This ranking method of Decision-making theory allows to detect the malware apps on mobile devices. The study performed by [19] suggests the probabilistic models to determine the set of relevant vulnerability detection rules and the geometric weighted average of frequency and impact scores of each threat in order to estimate their risk.

So, the topic of mobile apps security is broadly discussed by both industrial and scientific community. The given review of reports and research articles allows to state particular solutions of the mobile security problem have already been proposed. The considered works are mainly focused on separate threats detection and analysis. However, the authors do not pay much attention to the measurement of separate security threats identified in various models. Moreover, the problem of comprehensive assessment of the overall security based on the estimates of definite threats is not completely researched. In order to fill this gap, the given research considers the basic security detection principles and commonly known threats and proposes a new approach for estimation of mobile apps security level based on the fuzzy inference system (FIS) approach combined with the Analytical hierarchy process.

3 Proposed technique

In order to construct the list of most valuable mobile security threats, the Analytic hierarchy process (AHP) method was chosen. This method was developed by Saaty [20]. AHP is the most widely used decision-making methodology that researchers use in a huge number of fields [21]. Within this work, the AHP is applied to the matrix which was constructed from expert analysis data presented by [4] in the Top 10 Mobile Risks list shown in Table 1.

Insecure Authentication means that a mobile application contains security breaches in client-side or its server-side authentication mechanisms that can be employed by malicious users (attackers). Reverse Engineering means that the attacker gets the access to the app installed from the store and tries to investigate it using own software development tools. Insecure Data Storage occurs when mobile application stores critical or confidential customer information without encryption techniques (known as plain text); in this case, the third party could bypass system protections and obtain access to this private data. Code Tampering vulnerability may lead to serious sensitive data leakage, loss or system damage. Insufficient Cryptography could happen when developers use compromised or outdated encryption algorithms in their applications; an attacker could easily bypass them and the outcomes will be the same as storing sensitive data without any encryption. Improper Platform Usage leads to misuse of mobile operating system security controls. Insecure Communication contains a wide variety of potential threats. This condition could happen in case of improper configuration of encryption techniques (for example, SSL-certificates generation and validation), usage of outdated frameworks or libraries on a client and server-side, and enabling unnecessary access permissions, error logging or debug functions. By using this vulnerability, an attacker could obtain access to app databases and back-end servers. Client Code Quality exposes application to analysing, reverse engineering and source code modifying attacks and techniques. To avoid this, mobile applications must utilize solutions to detect source code modifications at runtime and prevent further access. The vulnerability could lead to data leakage as well as source code piracy. Extraneous Functionality assumes outside analysis of the mobile application and does not require involvement of end users.

Table 1. Security risks matrix according to [4].

Security risk	Exploit probability	Prevalence	Detection probability	Technical impacts
Insecure Authentication	EASY	COMMON	AVERAGE	SEVERE
Reverse Engineering	EASY	COMMON	EASY	MODERATE
Insecure Data Storage	EASY	COMMON	AVERAGE	SEVERE
Code Tampering	EASY	COMMON	AVERAGE	SEVERE
Insufficient Cryptography	EASY	COMMON	AVERAGE	SEVERE
Improper Platform Usage	EASY	COMMON	AVERAGE	SEVERE
Insecure Communication	EASY	COMMON	AVERAGE	SEVERE
Client Code Quality	DIFFICULT	COMMON	DIFFICULT	MODERATE
Extraneous Functionality	EASY	COMMON	AVERAGE	SEVERE

In general, the AHP method requires a following set of steps:

1. Firstly, presenting the original problem in the form of a hierarchical descending structure. On the top of it, there is the overall goal of the problem, the next level – various criteria, then sub-criteria. The lowest level of the hierarchy contains all alternatives [22].
2. Secondly, introducing experts opinions for paired comparisons at each hierarchical level.
3. Lastly, processing matrices of pairwise comparisons for finding local and global priorities [20].

To build comparisons matrix, a special scale of importance of weights of criteria/alternatives is used [20]. After the comparison of alternatives is completed, the weights of criteria importance should be calculated, similarly, by applying pairwise comparisons. The problem has N criteria and M alternatives. In the given case, it is necessary to construct N judgment matrices of $M \times M$ size and a single judgment matrix of $N \times N$ size. The final decision matrix of priorities A^i_{AHP} is determined by [23]:

$$A^i_{AHP} = \sum\nolimits_{j=1}^{n} a_{ij} w_j, i = 1,..., M. \quad (1)$$

The problem of defining security level of mobile applications cannot be properly described using Boolean logic truth-values (which may only be "true" or "false"). It will not allow to make any meaningful conclusions about further necessary steps to improve the security of the selected mobile application. In our case, more accurate assessments could be made by presenting the results mapped on a spectrum derived after reasoning from inexact or partial knowledge (for example, a list of security vulnerabilities provided in the research). Hence, fuzzy logic can be applied for describing the aforementioned problem.

For the implementation of the proposed model for estimation of the security of mobile applications, a Mamdani-type fuzzy inference system (FIS) was used. Figure 1 features a set of steps mandatory in the Mamdani algorithm [24].

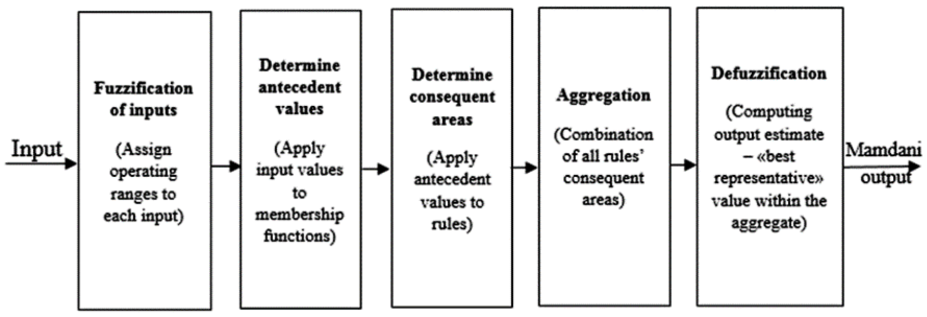
Input
Fuzzification of inputs
(Assign operating ranges to each input)
Determine antecedent values
(Apply input values to membership functions)
Determine consequent areas
(Apply antecedent values to rules)
Aggregation
(Combination of all rules' consequent areas)
Defuzzification
(Computing output estimate – «best representative» value within the aggregate)
Mamdani output

Fig. 1. Mamdani fuzzy logic steps schema

The problem should be presented as a set of rules:

$$F^{x=A}(AND/OR)^{y=B} THEN^{Z}, \quad (2)$$

where Z is resulting condition, for example, Z = "unsecure"; A, B are fuzzy inputs and Z is a fuzzy output.

At the rule evaluation stage, to estimate the disjunction and conjunction of rule antecedents the following formulae should be applied:

$$\mu A \cup B = \max[\mu A(x), \mu B(x)], \quad (3)$$

$$\mu A \cap B = \min[\mu A(x), \mu B(x)], \tag{4}$$

Among multiple defuzzification methods, in this work Centroid (centre of gravity - COG) method was chosen [25]. This method finds the point where a vertical line would split the aggregate set into two equal ones. The following formula is used to calculate Centroid of the fuzzy set A on the interval [a, b]:

$$COG = \frac{\int_a^b \mu A(x) X dx}{\int_a^b \mu A(x) dx} \tag{5}$$

The model requires a set of input variables, which are the corresponding numbers of most important security threats found by a mobile security analyst. An analyst should perform the manual security threats analysis of the source code and executable file of a mobile application for the presence of potential security threats, which description is presented in security reports such as Top 10 mobile risks [4].

4 Results

According to the expert analysis data provided by OWASP [4], a matrix which contains a list of security risks and their respective characteristics was constructed (Table 3.)

In order to define the most valuable threats the AHP method was applied and the following hierarchy was developed (Fig. 2).

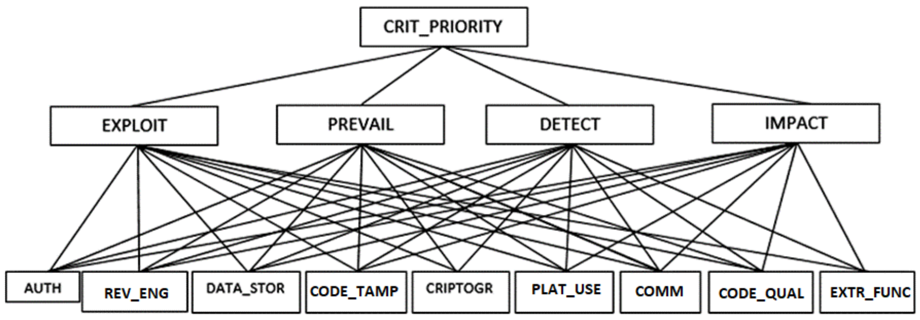

Fig. 2. AHP problem hierarchy

In this hierarchy the target is to define the priority of security criteria (CRIT_PRIORITY); criteria - Exploit probability (EXPLOIT), Prevalence (PREVAIL), Detection probability (DETECT), and Technical impacts (IMPACT). Finally, the alternatives are the security risks defined the OWASP list (Table 1). In case of this hierarchy, they are Insecure and Authentication (AUTH), Reverse Engineering (REV_ENG), Insecure Data Storage (DATA_STOR), Code Tampering (CODE_TAMP), Insufficient Cryptography (CRYPTOGR), Improper Platform Usage

(PLAT_USE), Insecure Communication (COMM), Client Code Quality (CODE_QUAL), and Extraneous Functionality (EXTR_FUNCT).

The next step is to construct matrices of pairwise comparisons for each security criterion. The intensity of importance was set according to the expert data (Table 3). The results of this step are presented in Tables 2-5.

Table 2. Matrices of pairwise comparisons for security criteria Exploit.

Security risk	Scores								
AUTH	1	1/5	1/5	1	1/5	1	1/5	1/5	1/5
REV_ENG	5	1	1	5	1	5	1	1	1
DATA_STOR	5	1	1	5	1	5	1	1	1
CODE_TAMP	1	1/5	1/5	1	1/5	1	1/5	1/5	1/5
CRYPTOGR	5	1	1	5	1	5	1	1	1
PLAT_USE	1	1/5	1/5	1	1/5	1	1/5	1/5	1/5
COMM	5	1	1	5	1	5	1	1	1
CODE_QUAL	5	1	1	5	1	5	1	1	1
EXTR_FUNC	5	1	1	5	1	5	1	1	1

Table 3. Matrices of pairwise comparisons for security criteria Prevail.

Security risk	Scores								
AUTH	1	5	1	1	1	1/5	1	1	1
REV_ENG	1/5	1	1/5	1/5	1/5	1/9	1/5	1/5	1/5
DATA_STOR	1	5	1	1	1	1/5	1	1	1
CODE_TAMP	1	5	1	1	1	1/5	1	1	1
CRYPTOGR	1	5	1	1	1	1/5	1	1	1
PLAT_USE	5	9	5	5	5	1	5	5	5
CONMM	1	5	1	1	1	1/5	1	1	1
CODE_QUAL	1	5	1	1	1	1/5	1	1	1
EXTR_FUNC	1	5	1	1	1	1/5	1	1	1

After performing corresponding AHP method calculations prior alternatives were found (Table 6).

Table 4. Matrices of pairwise comparisons for security criteria Detect.

Security risk	Scores								
AUTH	1	1	1	1	1	1	1	3	3
REV_ENG	1	1	1	1	1	1	1	3	3
DATA_STOR	1	1	1	1	1	1	1	3	3
CODE_TAMP	1	1	1	1	1	1	1	3	3
CRYPTOGR	1	1	1	1	1	1	1	3	3
PLAT_USE	1	1	1	1	1	1	1	3	3
COMM	1	1	1	1	1	1	1	3	3
CODE_QUAL	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1	1
EXTR_FUNC	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1	1

Table 5. Matrices of pairwise comparisons for security criteria Impact.

Security risk	Scores								
AUTH	1	4	1	1	1	4	1	1	1
REV_ENG	1/4	1	1/4	1/4	1/4	1	1/4	1/4	1/4
DATA_STOR	1	4	1	1	1	4	1	1	1
CODE_TAMP	1	4	1	1	1	4	1	1	1
CRYPTOGR	1	4	1	1	1	4	1	1	1
PLAT_USE	1/4	1	1/4	1/4	1/4	1	1/4	1/4	1/4
COMM	1	4	1	1	1	4	1	1	1
CODE_QUAL	1	4	1	1	1	4	1	1	1
EXTR_FUNC	1	4	1	1	1	4	1	1	1

Table 6. The AHP calculation results.

Alternative	Priority
DATA_STOR	0.1271
CRYPTOGR	0.1271
COMM	0.1271
PLAT_USE	0.124
CODE_TAMP	0.1049
AUTH	0.1049
CODE_QUAL	0.1022
EXTR_FUNC	0.1022
REV_ENG	0.08

The most important prior alternatives are Insecure Data Storage (DATA_STOR), Broken Cryptography (CRYPTOGR), Insecure Communication (COMM), and Improper Platform Usage (PLAT_USE).

Matrices of pairwise comparisons were constructed based on single expert opinions. In order to solve the assigned task, additional experts may be engaged, but there will be the problem of reconciliation of various experts' results. Besides, specifics of the task do not necessarily require different expert opinions that is why matrices were constructed using expert opinions obtained from one expert. Based on the found security alternatives it is now possible to proceed to the model construction step. To develop mathematical models using fuzzy logic, Mamdani algorithm was selected.

The model was constructed using the following list of fuzzy variables:

4. Let's consider the input fuzzy variables – DataStor, Comm, Cryptogr, and PlatUse – which define the number of security issues per the corresponding category (Insecure Data Storage, Insecure Communication, Insufficient Cryptography, and Improper Platform Usage).
5. Let's consider the output fuzzy variable SecurityLevel. This variable is used to describe the overall mobile application security level. According to obtained value from this variable, a security analyst can make further decisions about the overall mobile application security and necessity of additional actions.

6. The membership function has a trapezoidal form (trapmf). Membership Functions for Insecure Data Storage (DataStor), Insecure Communication (Comm) = Secure, Average, Unsecure.
7. Let's consider the membership function for SecurityLevel variable. Membership function has a trapezoidal form (trapmf). Membership Functions for Security Level (SecurityLevel) = lowest, bad, average, good, highest.
8. Let's consider the membership function for Cryptogr, PlatUse. Membership function has a trapezoidal form (trapmf). Membership Functions for Insufficient Cryptography (Cryptogr), Improper Platform Usage (PlatUse) = Secure, Average, Unsecure.

6) Let's consider the fuzzy rules defined in Table 7.

Table 7. Fuzzy rules for the model.

№	Rule
1	if (DataStor is **Secure**) and (Cryptogr is **Secure**) and (Comm is **Secure**) and (PlatUse is **Secure**) then (SecurityLevel is **highest**)
2	if (DataStor is **Secure**) and (Cryptogr is **Average**) and (Comm is **Secure**) and (PlatUse is **Secure**) then (SecurityLevel is **highest**)
3	if (DataStor is **Secure**) and (Cryptogr is **Secure**) and (Comm is **Average**) and (PlatUse is **Secure**) then (SecurityLevel is **highest**)
4	if (DataStor is **Average**) and (Cryptogr is **Secure**) and (Comm is **Secure**) and (PlatUse is **Secure**) then (SecurityLevel is **good**)
5	if (DataStor is **Average**) and (Cryptogr is **Secure**) and (Comm is **Average**) and (PlatUse is **Secure**) then (SecurityLevel is **good**)
6	if (DataStor is **Average**) and (Cryptogr is **Average**) and (Comm is **Average**) and (PlatUse is **Secure**) then (SecurityLevel is **average**)
7	if (DataStor is **Average**) and (Cryptogr is **Secure**) and (Comm is **Average**) and (PlatUse is **Average**) then (SecurityLevel is **average**)
8	if (DataStor is **Average**) and (Cryptogr is **Average**) and (Comm is **Secure**) and (PlatUse is **Average**) then (SecurityLevel is **average**)
9	if (DataStor is **Average**) and (Cryptogr is **Average**) and (Comm is **Average**) and (PlatUse is **Average**) then (SecurityLevel is **bad**)
10	if (DataStor is **Average**) and (Cryptogr is **Unsecure**) and (Comm is **Average**) and (PlatUse is **Average**) then (SecurityLevel is **bad**)
11	if (DataStor is **Average**) and (Cryptogr is **Average**) and (Comm is **Unsecure**) and (PlatUse is **Average**) then (SecurityLevel is **bad**)
12	if (DataStor is **Average**) and (Cryptogr is **Average**) and (Comm is **Average**) and (PlatUse is **Unsecure**) then (SecurityLevel is **bad**)
13	if (DataStor is **Secure**) and (Cryptogr is **Unsecure**) and (Comm is **Unsecure**) and (PlatUse is **Unsecure**) then (SecurityLevel is **bad**)
14	if (DataStor is **Unsecure**) and (Cryptogr is **Average**) and (Comm is **Unsecure**) and (PlatUse is **Secure**) then (SecurityLevel is **bad**)
15	if (DataStor is **Unsecure**) and (Cryptogr is **Unsecure**) and (Comm is **Unsecure**) and (PlatUse is **Unsecure**) then (SecurityLevel is **lowest**)

The following scale of security level (Table 8) was introduced to interpret the output model results properly.

Table 8. Scale of security level.

Security level	Description
1	Significantly risky
2	Risky
3	Lightly risky
4	Below average
5	Average
6	Above average
7	Good
8	Acceptable
9	Excellent

As a result, the model receives input variables (a number of security threats in the chosen mobile application) and after performing fuzzy logic calculations displays an output as an overall security level of the mobile app. Based on the model designed in the previous section, the real-time software has been built for defining security level of mobile applications. After that, a research on several open-source applications available on GitHub was conducted. The characteristics of the selected apps are presented in Table 9.

Table 9. List of mobile apps which security level are analyzed.

App name	Version	Description
TVHeadEnd	4.2	TVHeadEnd PVR client
Xabber Classic	0.9.31a	Instant messaging client
Shaarlier	1.6.1	Share links on Shaarli
TripleCamel	1.0.4	Check price history of Amazon products
AndroPTPB	1.1	Post snippets and files with the ptpb service
Quick Dice Roller	2.1.3	Flexible, complete and handy dice roller
Gobandroid	2.5.9	Ancient Go game

The analysis results (defined security level) were obtained using the software based on the developed model (Section 4). The number of security issues per each category and corresponding overall security level are presented in Table 10.

According to the provided results, it is possible to say that the mobile application “Gobandroid” is significantly risky and requires additional work to improve its security while the app “Shaarlier” could be safely used without additional security improvements.

Table 10. Results of defining security level for aforementioned apps.

App	Data storage	Cryptog-raphy	Commu-nication	Extraneous functionali-ty	**Security Level**
TVHeadEnd	5	5	5	5	**2.67 (Risky)**
Xabber Classic	7	6	6	7	**1.42 (Sign. risky)**
Shaarlier	1	1	1	1	**8.92 (Excellent)**
TripleCamel	2	1	3	1	**6.52 (Above avg.)**
AndroPTPB	2	4	3	1	**4.43 (Below avg.)**
Quick Dice Roller	1	4	3	1	**5.5 (Average)**
Gobandroid	7	8	7	8	**1.4 (Sign. risky)**
MineSweeper	6	7	7	7	**1.42 (Sign. risky)**

5 Discussion

The review of sources on the topic of mobile apps security showed that the problem of model design for the comprehensive assessment of the overall application security is still relevant and requires a solution. In this research, a model for defining overall application security level was proposed. This model was developed as a next step to the OWASP [4] research and was constructed using security issues data submitted by teams who made this research. Except that list, the model can also use results of the Mobile Application Security Study [12] and the research [6] to make more accurate encryption threats assessments. Security analysis tools presented by [13] and [7] can be used along with the proposed model for retrieving input data for overall mobile security level estimation. Instead of manual examination, a security analyst could use these tools to perform the necessary analysis and then use the obtained results as the model inputs which helps to receive a comprehensive representation of app security. Analysts can apply methodologies and techniques presented in [15] and [17] for various criteria testing. Their results can also be used as inputs for the provided model. The model is suitable for processing results obtained by using all three major testing methods – static, dynamic, and forensic methods [18].

The model is based on the third-party mobile security threats analysis of the OWASP Foundation [4], thus model results strictly depend on the correctness and completeness of the aforementioned research. Other research companies may publish different threats analyses in their papers which will require additional model alteration. Another limitation of the developed model is that it is based on fuzzy inference system, thus, the results may contain small imprecision. Nevertheless, a security analyst should have results that allow him to make further decisions about mobile application security and the model can successfully handle this task, possible fuzzy logic results inaccuracy should not affect an overall decision-making process.

6 Conclusions and future work

In this research, a model for estimating the security level of mobile applications has been proposed. A real-time software system has been developed to implement the suggested model. The experiments were conducted on a set of open-source mobile applications. This model also allows narrowing the range of potential vulnerabilities, which presence should be checked by the analyst in particular mobile app. This step allows reducing significantly the amount of work which mobile security specialist should perform to evaluate the level of mobile application security. It was achieved by defining which classes of security threats pose the most significant risks to the overall application's security level.

The main contributions of the paper are: 1) An efficient method for estimating the security level of mobile application. 2) A set of most valuable mobile security threats. 3) The developed real-time software system which allows performing a security testing of mobile applications. In future, it is planned to make several improvements to this model and software product. The model has a great reusability potential – it is possible to modify easily the classes of security threats, according to threats list updates published by different software security companies. Additionally, it is possible to improve functionality by adding scanning methods, which could perform checks of threat presence in a source code of mobile application and automatically calculate inputs for model without the manual work of a security analyst. Moreover, the efficiency of the model can be advanced by redefining the list of security issue categories according to new researches which will be published in future.

7 References

1. Saifi R The 2017 Mobile App Market_ Statistics, Trends, and Analysis. Retrieved from (2017) http://www.business2community.com/mobile-apps/2017-mobile-app-market-statistics-trends-analysis-01750346#op3a6wsaZb0h6d3T.97.
2. Beek C, Dinkar D, Gund Y, Lancioni G, Minihane N, Moreno F, Weafer V McAfee Labs Threats Report: June 2017. McAfee Labs Report, (June), pp 1–83 (2017) https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf.
3. Pradeo Pradeo's biannual mobile applications threats review for S1 (2017) https://www.pradeo.com/en-US/datasheet/mobile-applications-threats-review-S12017.
4. The OWASP Foundation Top 10 mobile risks (2016) https://owasp.org/www-project-mobile-top-10/, 03/05/2020
5. Girard J, Zumerle D, Reed B, Firstbrook P and Willemsen B Predicts 2017: endpoint and mobile security (2017) https://www.gartner.com/doc/3512932?ref=AnalystProfile&srcId=1-4554397745.
6. Kieseberg P, Frühwirt P, Schrittwieser S, Weippl ER Security tests for mobile applications — Why using TLS/SSL is not enough. 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 1-2 (2015) doi: 10.1109/ICSTW.2015.7107416
7. Morgado I, Paiva ACR Mobile GUI testing. Software Qual J 26, 1553–1570 (2018). https://doi.org/10.1007/s11219-017-9387-1.

8. Kaur S, Singh DK Mobile Application Testing based on Privacy information Encoding Mechanism (2016)
9. Vidal J M, Monge MAS, Villalba LJG A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences. Knowledge-Based Systems, 150, 198–217 (2018) https://doi.org/10.1016/j.knosys.2018.03.018
10. Asăvoae IM, Nguyen HN, Roggenbach M, Shaikh SA Software model checking: A promising approach to verify mobile app security-A position paper. In Proceedings of the 19th Workshop on Formal Techniques for Java-Like Programs, FTfJP 2017 - Co-located with ECOOP 2017. Association for Computing Machinery, Inc. (2017) https://doi.org/10.1145/3103111.3104040.
11. Hewlett Packard Enterprise Mobile application security report (2016) http://h22168.www2.hpe.com/docs/capgemini/Mobile%20Report%20ver%2010.2.pdf.
12. Hewlett Packard Enterprise (2013) Mobile application security study. 2013 report. http://files.asset.microfocus.com/4aa5-1057/en/4aa5-1057.pdf.
13. Zuo C, Wang W, Wang R, Lin Z Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services. Internet Society (2017) https://doi.org/10.14722/ndss.2016.23146.
14. Kim H, Yeo H, Hwang HJ, Ramos CH, Marreiros G. Effective Mobile Applications Testing Strategies (2016)
15. Meng H, Thing VLL, Cheng Y, Dai Z, Zhang L A survey of Android exploits in the wild. Computers and Security, 76, pp 71–91 (2018) https://doi.org/10.1016/j.cose.2018.02.019.
16. Netcraft Mobile app security testing (2018) https://www.netcraft.com/security-testing/mobile-app-security-testing/.
17. Wilhelm T Professional Penetration Testing: Creating and Learning in a Hacking Lab: Second Edition. Elsevier Inc. pp 1–445 (2013) https://doi.org/10.1016/C2012-0-00443-7
18. Stahl F, Stroher J Security testing guidelines for mobiles apps. (2013) https://www.owasp.org/images/0/04/Security_Testing_Guidelines_for_mobile_Apps_-_Florian_Stahl%2BJohannes_Stroeher.pdf.
19. Sadeghi A, Esfahani N, Malek S Mining mobile app markets for prioritization of security assessment effort. In WAMA 2017 Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics, pp. 1–7. Association for Computing Machinery, Inc. (2017) https://doi.org/10.1145/3121264.3121265.
20. Saaty TL Decision making — the Analytic Hierarchy and Network Processes (AHP/ANP). Journal of Systems Science and Systems Engineering, 13(1), pp 1–35 (2004) https://doi.org/10.1007/s11518-006-0151-5.
21. Vaidya OS, Kumar S Analytic hierarchy process: An overview of applications. European Journal of Operational Research, 169(1), pp 1–29 (2006) https://doi.org/10.1016/j.ejor.2004.04.028.
22. Kousalya P, Mahender Reddy G, Supraja S, Shyam Prasad V Analytical Hierarchy Process approach – An application of engineering education. Mathematica Aeterna, 2(10), pp 861–878 (2012) https://doi.org/10.1016/j.jvir.2016.06.036.
23. Triantaphyllou E, Mann SH Using the Analytic Hierarchy Process for Decision Making in Engineering Applications: Some Challenges. International Journal of Industrial Engineering: Theory, Applications and Practice, 2(No. 1), pp 35–44 (1995)
24. Fuzzy inference systems. http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm
25. Mamdani fuzzy model (2015) http://researchhubs.com/post/engineering/fuzzy-system/mamdani-fuzzy-model.html.