NETWORK AND APPLICATION SECURITY

DOI 10.20535/2411-1031.2020.8.1.218003
UDC 004[942::413.4]

MYKHAILO ANTONISHYN

MOBILE APPLICATIONS VULNERABILITIES TESTING MODEL

The process of testing vulnerabilities of mobile software applications has been analysed. This is due to the need to prevent violations of confidentiality, integrity and availability of information. Individual users and the state as a whole benefit from the preservation of these properties. However, in practice this is mostly neglected, and attention is paid to the functional testing. While the known approaches of testing vulnerabilities of the mobile software applications are focused on the study of certain aspects: either a server or a client. At the same time, the applicability of the international standards of testing vulnerabilities in mobile software applications has been established. A characteristic feature of their guidelines is the focus on OWASP methodology. It determines the rating of the most critical vulnerabilities, standard and test scenarios, tools for determining the level of security. They are summed up in OWASP Mobile TOP 10, OWASP MASVS, and OWASP MSTG recommendations. According to OWASP MSTG, vulnerabilities in mobile software apps are tested using OWASP MASVS. There are three parts in these documents, which are the following: general, Android, iOS. Also, these documents define common scenarios for each level of testing vulnerabilities in mobile software applications, as stated in MASVS. The level of security of mobile software applications is determined based on the results of the tests, namely: the test has been passed, the test has not been passed, and the test is not used for the mobile software application. However, the practical use of OWASP methodology is complicated by the focus on the client side of mobile software applications, the subjectivity of the choice of stages and their sequence. To prevent these limitations, a model for testing vulnerabilities in mobile software applications has been developed. A dependency graph is used to codify this procedure. This allows you to determine the stages of testing vulnerabilities in both client and server parts. In addition, it helps you to explain which testing stages to choose, their order, and the appropriate tools. This justification is accomplished by building a dependency relationship between them. An example of its formulation is "the execution of the next stage is preceded by the execution of the previous one". The obtained results are demonstrated in the example of SSL pinning vulnerability testing.

Keywords: mobile application, vulnerability, MASVS, OWASP, Android, vulnerabilities testing model, dependency graph.

Problem statement. Vulnerabilities in mobile software apps are tested to prevent data breaches in terms of confidentiality, integrity, and availability. Both the user and the organization, or the state as a whole, benefit from the preservation of these features [1] - [2]. However, in practice, this procedure is frequently ignored or given insufficient attention (for example [3], Fig. 1). This is due to the importance placed on meeting the functional requirements for mobile software applications.

Simultaneously, most methods for testing the vulnerabilities of mobile software apps are focused on a single aspect of their use [4]. For instance, on the client or on the server. Vulnerability statistics based on the outcomes of testing mobile software applications back this up (for example [5], Fig. 2). It's frequently followed by a proclivity to focus this strategy on the client. Furthermore, the arbitrary nature with which certain actions and the entire process are interpreted and implemented. As a result, vulnerability testing of mobile software applications is important.

Analysis of recent research and publications. Mobile software applications are tested according to the guidelines of international and national regulations [6] - [10]. A generalized

description of this process is given in [6]. While the features of testing vulnerabilities of mobile software applications are disclosed in [9], [10]. The methodological basis of these documents is the OWASP (Open Web Application Security Project) guidelines, namely [7], [8]: OWASP Mobile TOP 10, OWASP MASVS, OWASP MSTG. However, despite the general acceptance, their use in practice is limited by their focus on the client's side of the mobile software application. But at the same time, their use is also limited by the subjectivity of the choice of stages and their sequence in testing vulnerabilities.

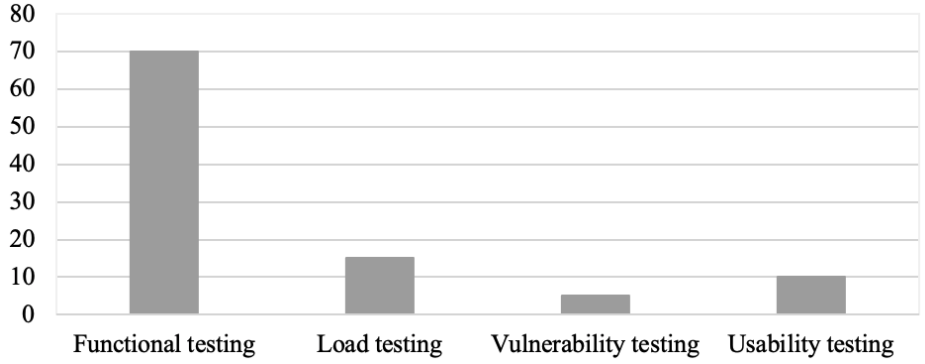

Figure 1 – Testing resources for mobile software apps are allocated based on their types

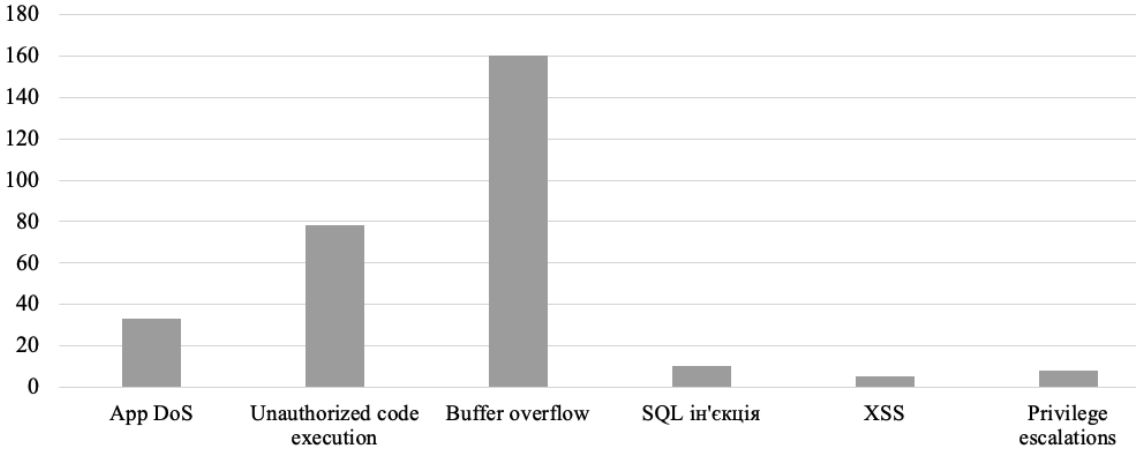

Figure 2 – Vulnerabilities of tested mobile software applications in 2019

Features of testing vulnerabilities of mobile software applications are considered in [11]. It has been formed the limitations of the following process. One of these limitations is the necessity to consider the details of vulnerability testing in the context of the mobile software application life cycle. [12], [13] discuss methods for testing mobile software vulnerabilities. Their use is based on the OWASP Mobile TOP 10 methodology. However, its practicality is limited by the focus on testing vulnerabilities in mobile software applications and information, mostly from the mobile device. The focus on testing vulnerabilities in mobile software applications and information, which comes mostly from the mobile device, limits its applicability. [14] - [16] provide the steps for evaluating mobile software vulnerabilities.

Therefore, the use of the results obtained in [11] - [16] is based on the OWASP guidelines. But at the same time, it is limited to a focus on the client sections of mobile software applications, subjectivity in the selection of vulnerability testing phases, and appropriate tools. The purpose of this research is to define the characteristics of mobile software application vulnerability testing.

The aim of this paper is research is to define the characteristics of mobile software application vulnerability testing.

The main material researches. The OWASP Mobile Security Project is a testing technique for mobile software applications' security. It identifies the severity of the most severe vulnerabilities, as well as standard and test scenarios and tools for assessing security. Mobile app developers and information security experts will benefit from this course. The OWASP Mobile Security Project's structure is interconnected; for example, the OWASP Mobile TOP 10 ranking of the most important vulnerabilities is based on tests conducted according to the testing standard, which is the Mobile Application Security Verification Standard. On the official OWASP page, however, this is not properly defined. To be more precise, only the scheme of interaction between the standard, testing scenarios and the tool for determining vulnerability assessments is given (for example [8], Fig. 3):

a) A checklist is a tool for calculating the amount of security + security, which is subsequently represented in graphs. This enables you to create zones with increased security.

b) Requirements - The Mobile Application Security Verification Standard (MASVS) is used to increase and verify the level of security of mobile software applications.

c) Test Cases - The Mobile Security Testing Guide (MSTG) defines standard tests for each stage that are described in MASVS and consists of common, Android, and iOS parts.

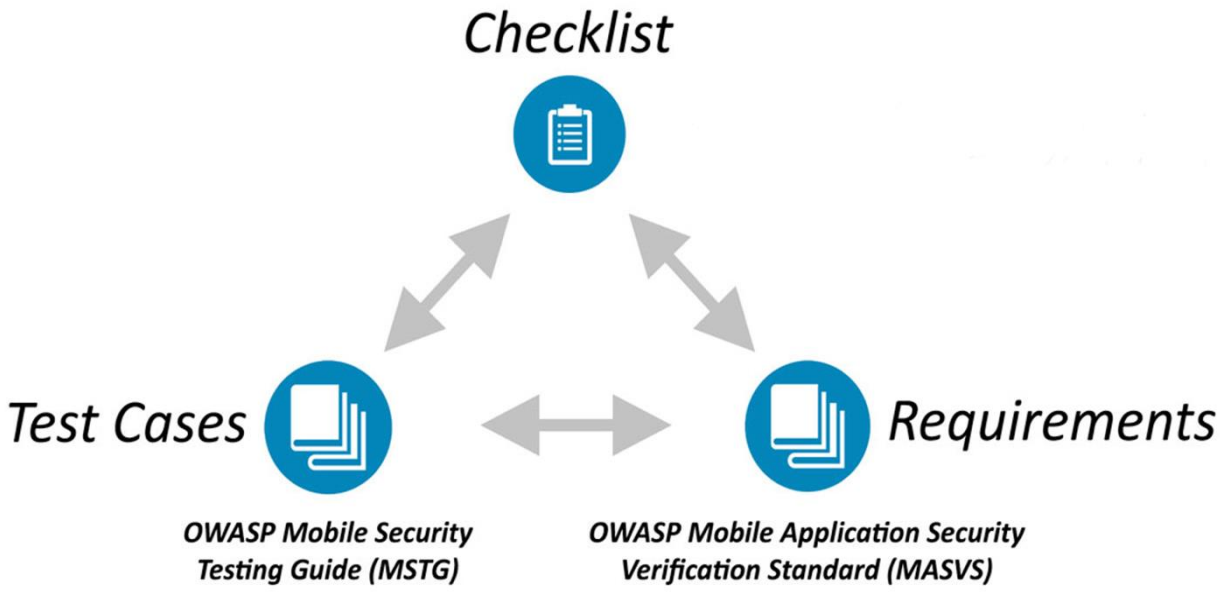

Figure 3 – Interaction between structural elements of OWASP Mobile Security Project

The level of security of mobile software applications is checked according to the OWASP MASVS standard (OWASP Mobile Application Security Verification Standard). It is used to both determine and verify the level of security. The standard was developed to achieve the following:

– the metrics;
– a basis for testing;
– a certification checklist.

OWASP MASVS standard defines two general levels of security and one additional one (for example, [8], Fig. 4). Each of them has its own set of tests, which is defined and described by the methodology of MSTG, which determines the methods and tools of testing.

To test within OWASP MASVS, MSTG testing guidelines are used. This document defines typical scenarios for each stage of testing mobile software vulnerabilities, which are described in MASVS, and consists of three parts:

– the general part describes vulnerability testing scenarios for mobile software applications running Android and iOS operating systems.

– the Android-part describes specific scenarios for testing vulnerabilities of mobile software applications running the Android operating system.

– the iOS part describes specific scenarios for testing vulnerabilities of mobile software applications under the control of the iOS operating system.

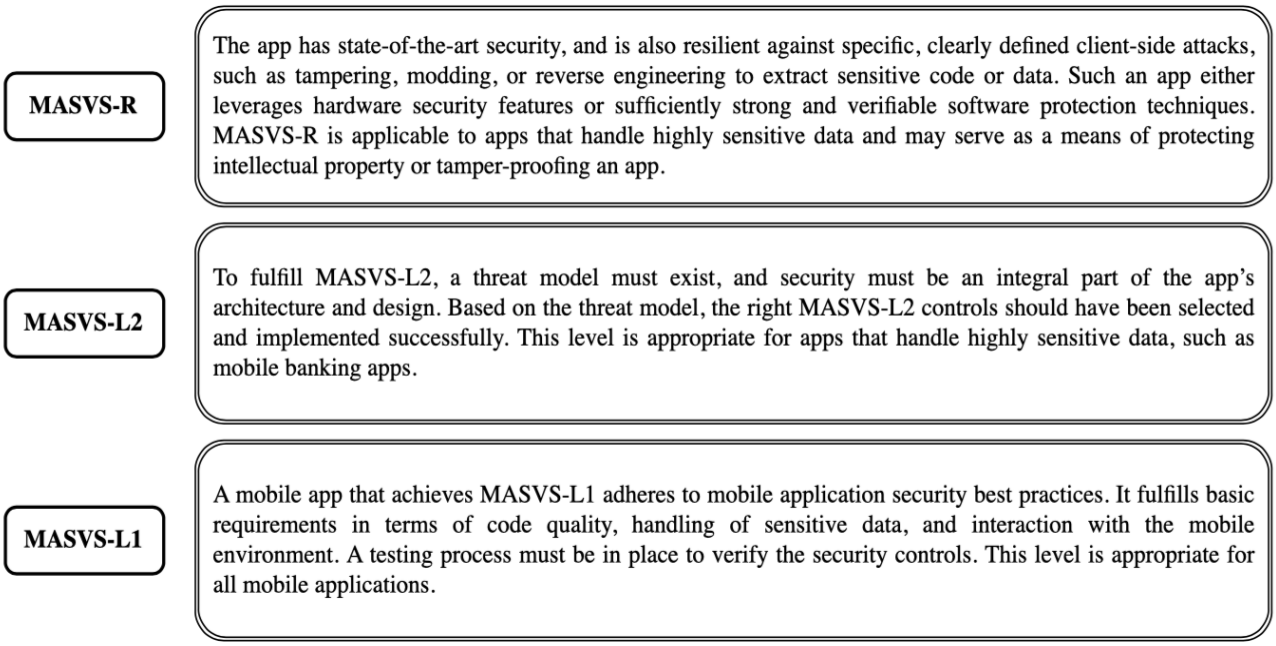

Figure 4 – Levels of vulnerability testing of mobile software applications according to MASVS

Testing of mobile software applications according to the MASVS standard takes place according to certain scenarios, which are described and constantly updated in the MSTG guidelines. According to the results of the tests, the level of ensuring their safety is determined. This gives one of the probable values, which are: T (test passed), F (test failed) and N/A (not used for mobile software application).

Each part is characterized by its own stages of testing. However, each stage is determined by a set of scenarios. Each scenario is performed by the established testing steps. Stages and scenarios can be performed independently of each other. Therefore, to perform comprehensive testing of mobile software applications, it is necessary to run all the scenarios with MSTG. In addition, it should be noted that typical scenarios are executed within the stages, but there are also specific ones. At the same time, typical scenarios have excellent steps, which depend on the settings of mobile operating systems and the skills of professionals. This is because each test can be performed in different ways and with different depth of study. For example, if you need to test the correctness of the implementation of SSL pinning, in a mobile software application, then this is a separate stage of testing, which is called Network Security API. Its implementation is possible in several different scenarios (see, for example [17], Fig. 5).

Fig. 5 shows examples of SSL pinning testing scenarios. They demonstrate that testing can take different and combined scenarios. First of all, there are steps to test the availability and functionality of SSL pinning. In addition, the steps that can be used depending on the tasks of testing, the experience of specialists. This example demonstrates that testing scenarios, their steps depend on the approach of testing, the level of training and the conditions under which testing is conducted. Additionally, it should be noted that each step of the testing scenario depends on the previous one. Therefore, the condition for switching between them is True or False. As a result, the presence of dependencies between stages (scenarios) allows to formalize the process of testing vulnerabilities of mobile software applications using dependency graphs [18].

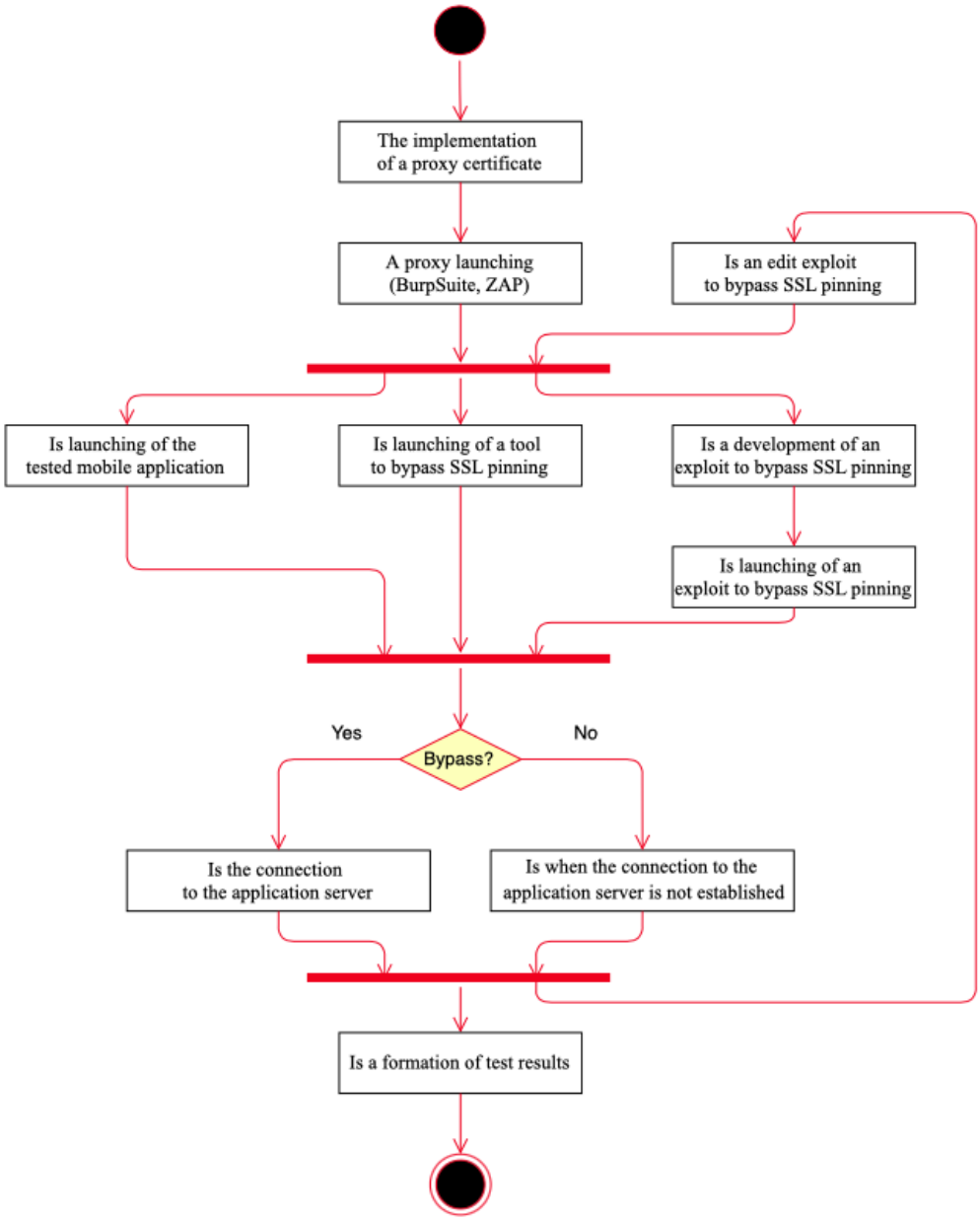

Figure 5 – Testing vulnerabilities of mobile software applications running Android operating system on the example of SSL pinning

A dependency graph is a directed graph that displays the relationship of multiple stages of vulnerability testing of mobile software applications with an established transitivity ratio (for example, "the next step is preceded by the previous one") between them. [18] The dependence graph is a graph:

$$G = \{V, T\},$$

where V means many stages of testing vulnerabilities of mobile software applications according to the OWASP;

R is a transitivity ratio, $R \subset V \times V$;

T is a transitive circuit R, $T \subseteq R$.

For example, to test an SSL pinning vulnerability, the set is specified by the following V elements:

$$V = \{v_i\},\ i = \overline{1,11},$$

$$R = \{(v_1; v_2), (v_2; v_3), (v_3; v_7), (v_7; v_9), (v_9; v_{11})\}, R \subset V \times V,$$

$$v_1 R v_2 \rightarrow v_2 R v_3 \rightarrow v_3 R v_7 \rightarrow v_7 R v_9 \rightarrow v_9 R v_{11}$$

where v_1 is the implementation of a proxy certificate;

v_2 is a proxy launching (BurpSuite, ZAP);

v_3 is launching of the tested mobile application;

v_4 is launching of a tool to bypass SSL pinning;

v_5 is a development of an exploit to bypass SSL pinning;

v_6 is launching of an exploit to bypass SSL pinning;

v_7 is SSL pinning checking;

v_8 is the connection to the application server;

v_9 is when the connection to the application server is not established;

v_{10} is an edit exploit to bypass SSL pinning;

v_{11} is a formation of test results.

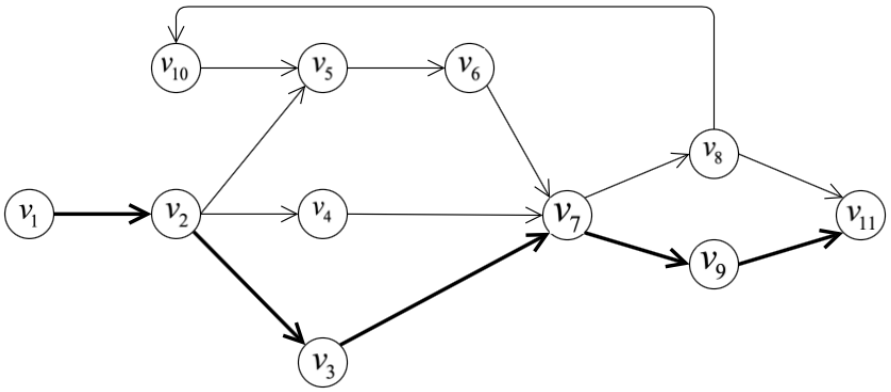

Figure 6 – The dependency graph of SSL pinning vulnerability testing steps

Conclusions. Therefore, based on the results of the analysis of existing standards for testing vulnerabilities of mobile software applications, their focus on the use of OWASP methodology has been established. This boils down to the implementation of the OWASP Mobile TOP 10, the OWASP MASVS and the OWASP MSTG guidelines. However, this approach is limited in practice by focusing on the client's side of mobile software applications, the subjectivity of the choice of stages and their sequence in vulnerability testing. To overcome these limitations and in particular to establish the features of this process, a model based on the dependence graph is proposed. Its use allows to formalize the process of testing vulnerabilities of mobile software applications. This is achieved by establishing a relationship between its stages, scenarios, steps. As an example, it is considered "the execution of the next stage is preceded by the execution of the previous one." The obtained results are demonstrated on the example of SSL pinning vulnerability testing.

REFERENCE

[1] International Organization for Standardization. (2011, Nov. 21). *ISO/IEC 27034-1, Information technology. Application Security*. [Online]. Avaliable: https://www.iso.org/standard/44378.html. Accessed on: Dec. 17, 2019.

[2] The President, the Prime Minister and the Ministry of Finance presented the mobile application "Diya". [Online]. Avaliable: https://www.kmu.gov.ua/news/prezident-premyer-ministr-mincifra- prezentuvali-mobilnij-zastosunok-diya. Accessed on: Dec. 17, 2019.

[3] A. Kramer, and B. Legeand, *Model-based testing essentials: Guide to the ISTQB Certified Model-Based Tester Foundation Level*. Hoboken, USA: Willey&Sons, Inc., 2016.

[4] M. Antonishyn, and O. Misnik, "Analysis of testing approaches to Android mobile application vulnerabilities", *Selected Papers of the XIX International Scientific and Practical Conference Information Technologies and Security*, vol. 2577, CEUR Workshop Proceedings, 2019, pp. 270-280. [Online]. Avaliable: http://ceur-ws.org/Vol-2577/paper22.pdf. Accessed on: Dec. 17, 2019.

[5] Quick heal annual threat report 2019. [Online]. Avaliable: https://www.google.com/url?sa=t&rct=Annual-Threat-Report-2019.pdf&usg=AOvVaBxp0Txjy0ExKPWN. Accessed on: Dec. 17, 2019.

[6] International Organization for Standardization. (2016, Okt. 05). *ISO/IEC 27034-6, Information technology. Security technique's Application Security, first edition*. [Online]. Avaliable: https://www.iso.org/standard/60804.html. Accessed on: Dec. 17, 2019.

[7] OWASP Mobile security testing guide (MSTG). [Online]. Avaliable: https://github.com/OWASP/owasp-mstg/. Accessed on: Dec. 17, 2019.

[8] OWASP Mobile application security verification standard (MASVS). [Online]. Avaliable: https://github.com/OWASP/owasp-masvs. Accessed on: Dec. 17, 2019.

[9] National Institute of Standards and Technology. (2019, Apr. 19). *NIST 800-163, Vetting the Security of Mobile application*. [Online]. Avaliable: https://doi.org/10.6028/NIST.SP.800-163r1. Accessed on: Dec. 17, 2019.

[10] National Information Assurance Partnership. (2019, Apr. 25). *Protection Profile for Mobile Device Fundamentals, Version 4.0*. [Online]. Avaliable: https://www.niap-ccevs.org/MMO/PP/pp_mdm_v4.0.pdf. Accessed on: Dec. 17, 2019.

[11] S. Zein, N. Salleh, and J. Grundy, "A systematic mapping study of mobile application testing techniques", *Journal of Systems and Software*, vol. 117, pp. 334-356, 2016, doi: 10.1016/j.jss.2016.03.065.

[12] S. Bojjagani, and V. N. Sastry, "STAMBA: Security Testing for Android Mobile Banking Apps", in *Advances in Signal Processing and Intelligent Recognition Systems. Advances in Intelligent Systems and Computing*, vol. 425, S. Thampi, S. Bandyopadhyay, S. Krishnan, KC. Li, S. Mosin, M. Ma, Berlin, Germany: Springer 2016, pp. 671-683, doi: 10.1007/978-3-319-28658-7_57.

[13] Z. Trabelsi, M. Al Matrooshi, and S. Al Bairaq, "Android based mobile apps for information security hands-on education", *Education and Information Technologies*, vol. 22, iss. 1, pp. 125-144, 2017, doi: 10.1007/s10639-015-9439-8.

[14] S. Roy, D. Chaulagain, and S. Bhusal, "Static Analysis for Security Vetting of Android Apps", in *From Database to Cyber Security. Lecture Notes in Computer Science*, vol 11170, P. Samarati, I. Ray, I.Ray, Berlin, Germany: Springer, 2018, pp. 375-404, doi: 10.1007/978-3-030-04834-1_19.

[15] T. Wu, X. Deng, and J. Yan, "Analyses for specific defects in android applications: a survey". *Frontiers of Computer Science*, vol. 13, iss. 6, pp. 1210-1227, 2019, doi: 10.1007/s11704-018-7008-1.

[16] V.-P. Ranganath, and J. Mitra, "Are free Android app security analysis tools effective in detecting known vulnerabilities?", *Empirical Software Engineering*, vol. 25, iss. 1, pp. 178-219, 2019, doi: 10.1007/s10664-020-09879-8.

[17] M. Antonishyn, "Four ways to bypass Android SSL. Verification and Certificate Pinning", in *Proc. VI International Scientific and Practical Conference Transfer of innovative Technologies*, Kyiv, 2020. pp. 96-98.

[18] M. Antonishyn, "The usage of dependency graphs to test the security of mobile software applications", in *Proc. Computer and information systems*, Kharkiv, 2020, p. 44, doi: 10.30837/ IVcsitic2020201369.

The article was received 30.03.2020.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] International Organization for Standardization. (2011, Nov. 21). *ISO/IEC* 27034-*1, Information technology. Application Security*. [Online]. Avaliable: https://www.iso.org/standard/44378.html. Accessed on: Dec. 17, 2019.

[2] The President, the Prime Minister and the Ministry of Finance presented the mobile application "Diya". [Online]. Avaliable: https://www.kmu.gov.ua/news/prezident-premyer-ministr-mincifra- prezentuvali-mobilnij-zastosunok-diya. Accessed on: Dec. 17, 2019.

[3] A. Kramer, and B. Legeand, *Model-based testing essentials: Guide to the ISTQB Certified Model-Based Tester Foundation Level*. Hoboken, USA: Willey&Sons, Inc., 2016.

[4] M. Antonishyn, and O. Misnik, "Analysis of testing approaches to Android mobile application vulnerabilities", *Selected Papers of the XIX International Scientific and Practical Conference Information Technologies and Security*, vol. 2577, CEUR Workshop Proceedings, 2019, pp. 270-280. [Online]. Avaliable: http://ceur-ws.org/Vol-2577/paper22.pdf. Accessed on: Dec. 17, 2019.

[5] Quick heal annual threat report 2019. [Online]. Avaliable: https://www.google.com/url?sa=t&rct=Annual-Threat-Report-2019.pdf&usg=AOvVaBxp0Txjy0ExKPWN. Accessed on: Dec. 17, 2019.

[6] International Organization for Standardization. (2016, Okt. 05). *ISO/IEC 27034-6, Information technology. Security technique's Application Security, first edition*. [Online]. Avaliable: https://www.iso.org/standard/60804.html. Accessed on: Dec. 17, 2019.

[7] OWASP Mobile security testing guide (MSTG). [Online]. Avaliable: https://github.com/OWASP/owasp-mstg/. Accessed on: Dec. 17, 2019.

[8] OWASP Mobile application security verification standard (MASVS). [Online]. Avaliable: https://github.com/OWASP/owasp-masvs. Accessed on: Dec. 17, 2019.

[9] National Institute of Standards and Technology. (2019, Apr. 19). *NIST 800-163, Vetting the Security of Mobile application*. [Online]. Avaliable: https://doi.org/10.6028/NIST.SP.800-163r1. Accessed on: Dec. 17, 2019.

[10] National Information Assurance Partnership. (2019, Apr. 25). *Protection Profile for Mobile Device Fundamentals, Version 4.0*. [Online]. Avaliable: https://www.niap-ccevs.org/MMO/PP/pp_mdm_v4.0.pdf. Accessed on: Dec. 17, 2019.

[11] S. Zein, N. Salleh, and J. Grundy, "A systematic mapping study of mobile application testing techniques", *Journal of Systems and Software*, vol. 117, pp. 334-356, 2016, doi: 10.1016/j.jss.2016.03.065.

[12] S. Bojjagani, and V. N. Sastry, "STAMBA: Security Testing for Android Mobile Banking Apps", in *Advances in Signal Processing and Intelligent Recognition Systems. Advances in Intelligent Systems and Computing*, vol. 425, S. Thampi, S. Bandyopadhyay, S. Krishnan, KC. Li, S. Mosin, M. Ma, Berlin, Germany: Springer 2016, pp. 671-683, doi: 10.1007/978-3-319-28658-7_57.

[13] Z. Trabelsi, M. Al Matrooshi, and S. Al Bairaq, "Android based mobile apps for information security hands-on education", *Education and Information Technologies*, vol. 22, iss. 1, pp. 125-144, 2017, doi: 10.1007/s10639-015-9439-8.

[14] S. Roy, D. Chaulagain, and S. Bhusal, "Static Analysis for Security Vetting of Android Apps", in *From Database to Cyber Security. Lecture Notes in Computer Science*, vol 11170, P. Samarati, I. Ray, I.Ray, Berlin, Germany: Springer, 2018, pp. 375-404, doi: 10.1007/978-3-030-04834-1_19.

[15] T. Wu, X. Deng, and J. Yan, "Analyses for specific defects in android applications: a survey". *Frontiers of Computer Science*, vol. 13, iss. 6, pp. 1210-1227, 2019, doi: 10.1007/s11704-018-7008-1.

[16] V.-P. Ranganath, and J. Mitra, "Are free Android app security analysis tools effective in detecting known vulnerabilities?", *Empirical Software Engineering*, vol. 25, iss. 1, pp. 178-219, 2019, doi: 10.1007/s10664-020-09879-8.

[17] M. Antonishyn, “Four ways to bypass Android SSL. Verification and Certificate Pinning”, in *Proc. VI International Scientific and Practical Conference Transfer of innovative Technologies*, Kyiv, 2020. pp. 96-98.
[18] M. Antonishyn, “The usage of dependency graphs to test the security of mobile software applications”, in *Proc. Computer and information systems,* Kharkiv, 2020, p. 44, doi: 10.30837/ IVcsitic2020201369.

МИХАЙЛО АНТОНІШИН

МОДЕЛЬ ТЕСТУВАННЯ УРАЗЛИВОСТЕЙ МОБІЛЬНИХ ПРОГРАМНИХ ЗАСТОСУНКІВ

Проаналізовано процес тестування уразливостей мобільних програмних застосунків. Це обумовлено необхідністю унеможливлення порушень конфіденційності, цілісності та доступності інформації. Збереженість даних властивостей важлива як для окремих користувачів, так і держави загалом. Однак, на практиці здебільшого цим нехтують і приділяють увагу функціональному тестуванню. Тоді як відомі підходи до тестування уразливостей мобільних програмних застосунків орієнтовані на дослідження окремих аспектів: або серверний, або клієнтський. Водночас встановлено застосовність міжнародних стандартів тестування уразливостей мобільних програмних застосунків. Характерною особливістю їх настанов є орієнтованість на методологію OWASP. Нею визначаються рейтинг найбільш критичних уразливостей, стандарт і сценарії тестування, інструментальні засоби визначення рівня забезпечення безпеки. Вони узагальнюються настановами OWASP Mobile TOP 10, OWASP MASVS, OWASP MSTG. Уразливості мобільних програмних застосунків тестуються у межах OWASP MASVS відповідно до OWASP MSTG. Даними документами визначаються типові сценарії для кожного етапу тестування уразливостей мобільних програмних застосунків, які описані в MASVS, та виокремлюються три частини: загальна, Android, iOS. За результатами проведених тестів визначається рівень забезпечення безпеки мобільних програмних застосунків, а саме: тест пройдено, тест не пройдено та не використовується для мобільного програмного застосунку. Однак, використання методології OWASP на практиці ускладнюється орієнтованістю на клієнтську частину мобільних програмних застосунків, суб’єктивністю обирання етапів і їхньої послідовності. Для запобігання цим обмеженням розроблено модель тестування уразливостей мобільних програмних застосунків. Даний процес формалізовано використанням графу залежностей. Це дозволяє визначати етапи тестування уразливостей як клієнтської, так і серверної частин. До того ж обґрунтувати обирання етапів тестування, їхньої послідовності та відповідних інструментальних засобів. Таке обґрунтування досягається встановленням умови залежності між ними. Як приклад її формулювання розглянуто “виконанню наступного етапу передує виконання попереднього”. Отримані результати продемонстровано на прикладі тестування уразливості SSL pinning.

Ключові слова: мобільний програмний застосунок, уразливість, MASVS, OWASP, Android, модель тестування уразливостей, граф залежностей.

Antonishyn Mykhailo, postgraduate student, Pukhov institute for modeling in energy engineering of National academy of sciences of Ukraine, Kyiv, Ukraine.
ORCID: 0000-0002-2665-0066.
E-mail: antonishin.mihail@gmail.com.
Антонішин Михайло Васильович, аспірант, Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України, Київ, Україна.