International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 18 (2018) pp. 13603-13609

A Conceptual Exploration for the Safe Development of Mobile Devices Software Based on OWASP

Celio Gil[1], Luis Baquero[1], Miguel Hernández[1] and Jesús D. Rodríguez[1]

[1] *Fundación Universitaria Los Libertadores, Bogotá D.C., Colombia.*

Abstract

The main objective of this article is to highlight the importance of guidelines in the development (software methodologies) of OWASP insurance for mobile applications, where security is conceived as a desirable property in the assurance of the quality of a secure mobile software product. Currently, the overcrowding of devices has allowed security to be recognized by protections and mitigations of risks at the system level and its architecture, and implemented during the development of applications and their operation. Thus, the OWASP safe mobile development guidelines are considered, which are linked to the top 10 security vulnerabilities and their implementation according to the most affected functionalities such as user authentication and password management, obfuscation. Of code, the control of payments and the storage of information among others.

According to the trends outlined, it is considered to be before the emergence of what some researchers call security engineering, as a complement to the safe software engineering, whose scope includes, among others, safety requirements, the safety model and the development of secure software; whose main objective as a research field is the production of techniques, methods, processes and tools that integrate the principles of safety engineering and quality, and that allow software developers to analyze, design, implement, test and deploy systems of secure software**.**

Keywords: Information Security, Methodologies, Mobile Devices, OWASP, Secure Software.

INTRODUCTION

Due to the great boom and growth of mobile applications, security is conceived as a desirable property for insurance and is part of the set of attributes to be considered in order to determine the quality of a software product. Currently, the overcrowding of devices has allowed security to be recognized by protections and mitigations of risks at the system level and its architecture, and implemented during the development of applications and their operation. Also because mobile devices are increasingly used in everyday life and consequently this increase is also reflected in the demand for application development, to illustrate the above, Figure 1 shows the increase in mobile data traffic [1].

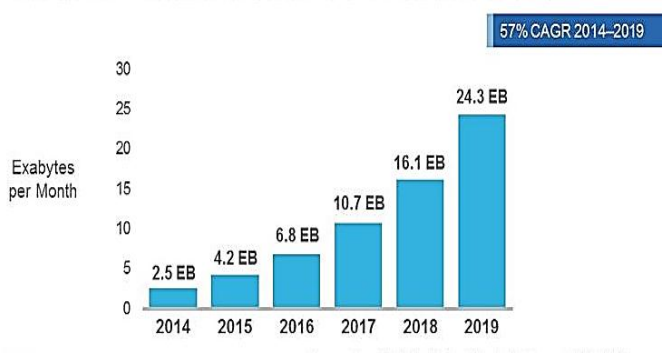

Figure 1. Growth of mobile global traffic (CISCO, 2017).

In the same way and contributing to what has been indicated above, new trends that boost the use of mobile devices are also highlighted, based on the study carried out by the IBSG (Internet Business Solutions Group) of CISCO [2]. which shows the trend towards increasing the use of mobile devices in organizations, this trend known as BYOD (Bring your own device) which in Spanish means "Bring your own device", shows that companies expect this behavior to increase with work purposes [2] as indicated in figure 2.

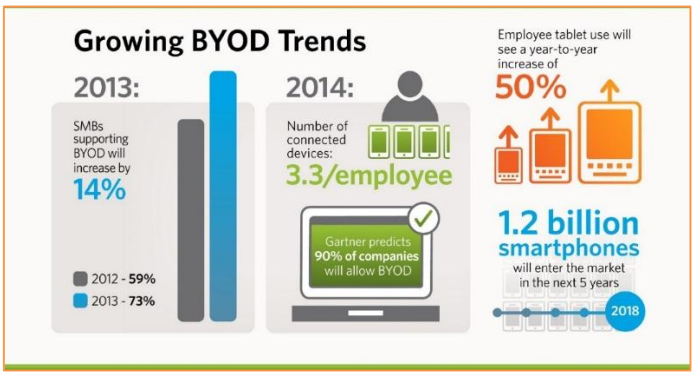

Figure 2. Projection of companies that expect the increase of BYOD in the coming years by country (Corey, 2014).

Taking into account the above and observing the proliferation of the use of these devices, thus, attacks on this type of terminals have increased, for which figure 3 shows the distribution of threat types [16], being its most representative item RiskTool, where this malware has the function of hiding files in the system, ending running processes and hiding applications.

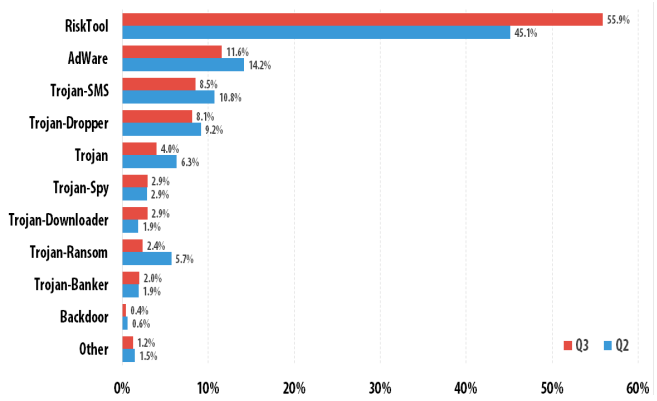

Figure 3. Distribution by types of malware on mobile devices (SsecureList, 2016).

As previously stated, it is identified that these malicious software called malware take advantage of weaknesses in both the operating system and architecture applications, because in some cases security guidelines are not applied and vulnerabilities are not evaluated. which may be exposed the application developed or to be developed [3], according to this and as an objective of this article will provide the indications to apply the mobile development guidelines OWASP Insurance (The Open Web Application Security Project) in the most relevant functionalities according to the top 10, which is periodically being updated, because this community is specialized in the survey of vulnerabilities, security tests for different technological platforms and where its top 10 vulnerabilities is one of the most important references important in the field of security for this type of equipment and other technologies

SAFE SOFTWARE DEVELOPMENT METHODOLOGIES

Software is typically an intangible element and that is why Systems Engineering has evolved developing publications, techniques and methods oriented to the construction of secure systems. All this has conformed the "Engineering of Safe Systems", a discipline that deals to the construction of systems that must remain functioning as expected in the face of badness, error, or chance [16]. Like any discipline, it focuses on the instruments, processes, and methods used to design, implement, and test complete systems, and adapt existing systems to their environment.

The systems engineering activities involved in the construction of secure systems are: (1) The development of systems security requirements; (2) The development of safe system designs; (3) The integration of subsystem components; (4) System tests and subsystems [16].

On the other hand, there are several methodologies that establish a series of steps in search of a more secure software capable of resisting attacks. These include Correctness by Construction (CbyC), Security Development Lifecycle (SDL), Cigital Touchpoints, Common Criteria, Comprehensive, Lightweight Application Security Process (CLASP), TSP-Secure.

The agile methods allow to have an iterative development, with continuous delivery cycles, a permanent contact with the client; allowing the risk managers of the company, certifiers and personnel responsible for security policies to be included as part of the stakeholders [17].

Nevertheless; the mentioned methodologies tend to focus on improving software quality, reducing the number of defects and complying with the specified functionality [18]; but nowadays, it is also necessary to deliver a product that guarantees a high degree of security [19].

MODELING SECURITY THREATS

To carry out the analysis and threat assessment, different models can currently be used, such as the SDL threat modeling tool (Security Development Lifecycle) [4], where this tool allows developers identify and mitigate security flaws. Figure 4 describes the cycle used by this tool.

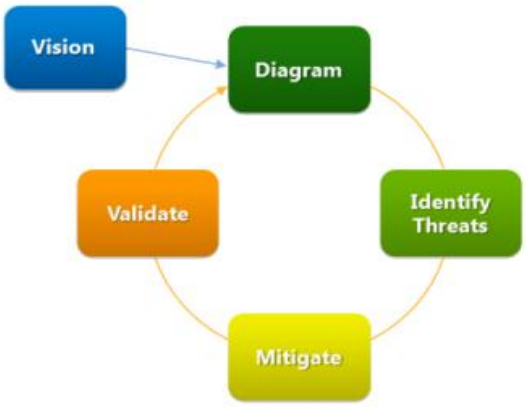

Figure 4. SDL threat modeling process (Microsoft, 2016).

In the same way, OWASP proposes a threat analysis modeling which should be applied throughout the life cycle of application development, since to the extent that this cycle is being executed, it is examined, identified and mitigated. Threats or vulnerabilities found [5].

During the modeling of these threats there are three high-level steps that must be carried out for the implementation of this model, which are:

• **Decompose the application**: that is, a sweep must be done in the application, know it and identify the relationships or interactions that this has with the outside and with the environment in which it will be found, the information inputs and outputs, which types of data will manage and how is the management of information and storage, this will identify several threats and then categorize them and give them a hierarchy.

• **Give hierarchy to threats**: once identified the different threats it is important to assign them a hierarchy, which can be done using different models such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) in English) [6] which helps

identify threats in the components of a system at the level of the attacker depending on their categories, allowing the creation of attack trees [7], which help identify threats and causes of each of these; You can also use the DREAD model (Damage, Reproducibility, Exploitability, Affected users, Discoverability) which helps to weigh the threats identified based on their risk [7], another model to give hierarchy to threats and its risks is OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation of its acronym in English) which can be an option when assessing threats and their risks, since it validates according to availability, confidentiality and types of active affected (Active: any information or element related to it) for this case the mobile application developed or developed [8].

• **Mitigate the threats**: after assigning the hierarchy to the threats, a mitigation map is made where the threat can be assigned and the action taken to mitigate it.

It is important to highlight that each of these steps must be documented and communicated with the interested parties, indicating how the mitigations are carried out, and the follow-up to identify the status of each one of them.

As we have seen above, these steps help to give the application not only security at the code level, but also in each of the stages of the application's life cycle, this same technique has been applied to identify the vulnerabilities in the mobile devices by OWASP which are shown below.

OWASP SECURITY VULNERABILITIES

As indicated above, the top 10 of OWASP is a summary of vulnerabilities found in mobile applications, which are illustrated in Table1:

Table 1. Top 10 vulnerabilities (OWASP, 2016).

Vulnerabilities	Description
M1 - improper use of the platform	This category covers the improper use of the platform or the non-use of security controls, platform permissions, undue use of TouchID, password services (KeyChain IOS) or some other security control that is part of the mobile operating system.
M2 - Unsafe data storage	This new category is a combination of M2 + M4 from Mobile Top Ten 2014. This covers unsafe storage of data and unwanted data leaks.
M3 - Insecure Communication	This covers poor link protocols, incorrect SSL versions, weak negotiation, unencrypted communication of sensitive data, etc.
M4 - Unsafe Authentication	This category captures the notions of end-user authentication or incorrect session management. This may include: • Do not identify the user at all when necessary • Failure to maintain the user's identity when required • Weakness in handling sessions
M5 - Insufficient cryptography	The code applies cryptography to a sensitive information asset. However, cryptography is insufficient in some way. Keep in mind that everything and everything related to TLS or SSL goes on M3. Also, if the application does not use cryptography at all when it should, it probably belongs to M2.
M6 - Insecure Authorization	This is a category to capture any authorization failure (for example, authorization decisions on the client side, forced navigation, etc.). It is different from authentication problems (for example, device registration, user identification, etc.).
M7 - Quality of the customer code	This would be the capture of all the implementation problems at the code level in the mobile client. This is different from server encoding errors. This would capture things like buffer overflows, format string vulnerabilities and various other code-level errors where the solution is to rewrite some code that is running on the mobile device.
M8 - Code adulteration	This category covers binary patches, modification of local resources, method hooks, method swizzling and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can directly modify the code, dynamically change the contents of the memory, change or

Vulnerabilities	Description
	replace the APIs of the system used by the application or modify the data and resources of the application.
M9 - Reverse Engineering	This category includes the analysis of the final binary kernel to determine its source code, libraries, algorithms and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker a view of the application's internal operation. This can be used to exploit other nascent vulnerabilities in the application, as well as to reveal information about back end servers, cryptographic constants and figures, and intellectual property.
M10 - Strange Functionality	Often, developers include hidden backdoor functionality or other internal security controls that are not intended to be put into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid application. Another example includes disabling two-factor authentication during the test.

Within the security vulnerabilities shown in Table 1, the most relevant vulnerabilities have been taken, where the application of the OWASP guidelines according to the following items is described.

Authentication and password management

The authentication in the applications is the initial interaction of the user with the application, where the identity of the user or other functionalities to be authenticated is verified, it may be at first glance a part of the software that does not have much complexity, but that It gives access to the operation of the application and relates various components for the management of it, as is the case of the permissions and privileges that the authenticated user has as well, such as the provision of functionalities for its use where the most important thing is Protect this characteristics as indicated below [9]:

1. Sometimes the application may require the user to create their password or pattern, against this it is advisable not to use the pattern due to its restrictions, since its length is not greater than 9 digits, which results in a number of low combinations and therefore not very strong, so it is very easy to validate each of these in a very short time. Another limitation is that each point can only be used once, in the same way the pattern can't omit an intermediate point on a line, just as the finger can't leave the touch surface, users usually use these patterns with combinations quite predictable [10] [11], which is why instead it is suggested that a password be created that follows a secure password policy, i.e. alphanumeric combinations with special characters and defined length, which will make this functionality of the application is more robust and secure [11].
2. Several mobile devices offer the possibility of using a password PIN, where this type of authentication does not have a very strong security, although it is based on the entry of a 4-digit number resulting in a number of combinations not very strong, which could be deciphered in a short time [10] [11], in its replacement it is advisable to use or create a password with secure password policies, although the trends of these types of authentication are focused on biometric devices [12].
3. It is also important to provide users with information on the strength of passwords during their creation, as this will indicate to the user the degree of security of their password [11].
4. 4. As a good practice, it is important to perform the validation of an additional parameter or to implement a two-step authentication policy since this will provide greater security when creating users and entering applications [11].
5. In order to verify the validation functionality of password strength, the tool indicated in [13] is recommended. Because it allows you to implement a secure password policy, and in addition to validating the strength, it indicates the execution time for the password entered.
6. Another important point in the security of the authentication and management of passwords is to validate at all times the data entered by the user by verifying the entered values, since in these fields cases of injection of database scripts can be presented and the rest.

The obfuscation of code

Another guideline of OWASP is to obfuscate the application code, since this will prevent reverse engineering of the application, preventing the source code from being obtained and modifications being made after its publication.

The obfuscation of code is the process that transforms the source code or intermediate code to make it more difficult to analyze [14], then indicate the points that are part of this guideline:

1. Try to obfuscate the application code whenever possible, through the execution of an automated code obfuscation software (commercial or open source), which is available in most development IDEs.

2. Apply anti-purification techniques, this section is quite simple since in most cases you should avoid that the debugger adheres to running processes, where with only a few configuration lines in the manifests of the application you can mitigate this vulnerability.
3. One of the most used tools to perform code obfuscation is ProGuard [15], this easy-to-use tool allows obfuscation of our code and helps us to eliminate all unused and variable classes without use, as well as to modify the names of the variables and methods.
4. Currently in the operating systems of mobile devices is already implemented ASLR "Address Space Layout Randomization" which is the random storage system of execution in memory and fulfills the function of assigning dynamic routes for the execution of applications. This technique is widely used by manufacturers of mobile devices such as Apple and Android and it is advisable during the design stage to identify if the application uses shared components, since this policy must be specified so that the components of these are used by other processes of other applications or the operating system.

Communication security

One of the most important factors of security in mobile devices is the security of communications, since it is the means by which the application interacts with the internet or allows the execution of external content, in this case the application has a browser internal to show the content, taking as reference the above listed security considerations for this section:

1. In case the application makes shipments or requests for information to external services it is advisable to encrypt the information of the packages sent and perform the corresponding decryption on the server side and vice versa in the case of reception of server-side responses.
2. In cases where the application has the functionality of web browsing it is advisable to validate the security certificates of the pages shown in order to validate the integrity of the same, this can be done by validating the dates of the certificates of the sites.
3. Validate that all requests and resources of the pages use SSL and that their internal elements such as images and styles apply this same security, since in case of not using it these elements may contain unsafe information and where this information should not be shown.

Storage and data protection

The information stored in the device is quite important and must comply with some characteristics since all the data handled by the application should not be stored in it, because these are very prone to damage or theft, taking into account the above, the following considerations for handling data in mobile applications:

1. Store information that is expressly necessary, at no time store data of passwords or sensitive data such as credit card information and others.
2. Classify the sensitive information that the application will handle, this classification will help you identify possible data that you must strictly store locally and also what information can be stored remotely.
3. Set the data deletion time in the application cache, since sometimes some applications use this temporary data to obtain access data and others.
   In case the sensitive information should be stored, it must be encrypted, and in case this information is temporary or temporary be sure to delete it as soon as the execution of the application ends.
4. In case of loss or theft of the device it is important to implement the erasure of the information remotely, where it will prevent the information from being left in inappropriate hands.
5. Instead of using sensitive data such as bank accounts you can make use of unique identifiers known between both parties (application and server), the above for applications with external interaction, which can only be interpreted by both parties.
6. Do not use the device identification number as the identifier of temporary files, it is advisable to use a random number or GUID instead.

Payment control

One of the most important roles in mobile applications and that are repeatedly used as a complement to these, are purchases through this medium, this type of transaction varies depending on the platform and architecture, since there are several methods for perform this task and for the case of this document will indicate the controls that must be taken into account to make this functionality the safest:

1. Warn the user and obtain financial consent and the consequences of using the application.
   2. Validate the location of the connection as this will allow you to identify if there are drastic changes in this and thus perform additional validations.

Session management

Within the good practices of development, is the administration of sessions, which is very important since it allows the use of data in a global way that can be used at any time, then the suggestions of good development practices are indicated for this section:

1. It is advisable to validate the session in each action of the application, this will validate the state of the same identifying whether the session expired or not and thus, prevent the application is still active and in some cases the user should be sent to the authentication interface.

2. In the same way that when validating the session in each action of the application, it is advisable to configure the expiration time of the same with a short or adequate time as indicated in ISO/IEC 27002 [16] .

CONCLUSIONS

As a result of the elaboration of this article, the importance of the implementation of safe development guidelines in mobile applications was identified, since several gaps are identified that are not visualized at first sight but through the analysis in each of the stages of the development was evidenced, it was also possible to verify how the use of mobile devices has been behaving at present, the trends that contribute and make the use of this type of hardware increase every day, where its use is not only personal, It is also projected as an instrument with high growth in companies and organizations, although the use of the desktop is still relevant, mobile devices will offer a large number of efficient uses, which will contribute to productivity and agility in the processes of each organization.

According to the trends outlined, it is considered to be before the emergence of what some researchers call security engineering, as a complement to the Safe Software Engineering, whose scope includes, among others, the security requirement engineering, the model of security and the development of secure software. Its main objective as a research field is the production of techniques, methods, processes and tools that integrate the principles of safety engineering and quality, and that allow software developers to analyze, design, implement, test and deploy systems of secure software. This new area of Engineering offers several advantages, among which are: 1. Allow the development of better techniques related to safety and better definitions of methodological work schemes; 2. Offer the basis for a complete and recognized security ontology, which allows developers to consider not only the technological challenges related to security, but also the social implications derived from them.

It is important to understand that, just as technology is constantly evolving, so cybercrime techniques are evolving, it should also be noted that the revision of vulnerabilities is not a static task over time, but instead is a dynamic and evolving task, because with each advance the degree of security must be compared to the existing and new features that will be part of the application.

Similarly, an analysis was made of the different considerations and methods to be used in the development of applications for mobile devices, it is identified that these methodologies and guidelines used in an indicated way can give mobile applications not only security in coding but also in the whole life cycle, since this type of responsibility does not fall directly into the development area, but rather part of the conception, analysis and design of the software, since from this point these considerations must be present, when applying the methodologies to identify threats and classify them, to measure the level of risk of each of them, and thus define the development guidelines necessary to make the developed application not only safe but also competitive in the market for the quality in each one of the steps of its life cycle.

REFERENCES

[1] Mohamed Ghallali, E. B. (2012). The safety of mobile phones: The methods of preventing the spread of malware.

[2] http://biblioteca.libertadores.edu.co:2087/xpls/icp.jsp?arnumber=6481989#fig_2.

[3] Cisco IBSG, 2. (2012). BYOD: a global perspective. Homepage, http://www.cisco.com/c/dam/en_us/about/ac79/docs/re/byod/BYOD_Horizons-Global_LAS.pdf

[4] SEGURINFO. (25 de 10 de 2016). Los 10 errores mas comunes de seguridad de aplicaciones móviles. Obtenido de http://www.segurinfo.org/detalle.php?a=los-10-errores-mas-comunes-de-seguridad-de-aplicaciones-moviles&t=2&d=542

[5] Microsoft. (2016). Microsoft Security Development Lifecycle (SDL) – Process Guidance. https://msdn.microsoft.com/es-es/library/windows/desktop/84aed186-1d75-4366-8e61-8d258746bopq.aspx.

[6] OWASP. (2016). Modelado de Amenazas. Obtenido de https://www.owasp.org/index.php/Modelado_de_Amenazas

[7] Shostack, A. (2014). Threat Modeling: Designing for Security. En A. Shostack, Threat Modeling: Designing for Security (págs. 61-86). Wiley.

[8] OWASP. (2016). Threat Risk Modeling. Obtenido de https://www.owasp.org/index.php/Threat_Risk_Modeling#DREAD

[9] MINTIC. (6 de 11 de 2016). Guia Seguridad informacion Mypimes. Obtenido de http://www.mintic.gov.co/gestionti/615/articles-5482_Guia_Seguridad_informacion_Mypimes.pdf

[10] José A. Montoya S, Z. R. (2012). GESTIÓN DE IDENTIDADES Y CONTROL DE ACCESO DESDE UNA. Obtenido de http://web.usbmed.edu.co/usbmed/fing/v3n1/v3n1a3.pdf

[11] ESET-LA.COM. (01 de 11 de 2016). WELIVESECURITY. Obtenido de ¿Cuán difícil es descubrir el patrón de desbloqueo en Android?: http://www.welivesecurity.com/la-es/2015/09/07/descubrir-patron-de-desbloqueo-en-android/

[12] ESST-LA.COM. (13 de 06 de 2014). La matemática de las claves: ¿numérica o alfanumérica? Obtenido de http://www.welivesecurity.com/la-es/2014/06/13/matematica-claves-numerica-alfanumerica/

[13] OWASP. (18 de 07 de 2016). Proyecto OWASP Mobile Security. Obtenido de

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

[14] KAPERSKY. (2016). Password Cheker. Obtenido de https://password.kaspersky.com/es/

[15] IEEE, S. A.-S.-P.-M.-T. (29 de 02 de 2016). A study & review on code obfuscation. Obtenido de http://biblioteca.libertadores.edu.co:2087/xpls/icp.jsp?arnumber=7583913

[16] www.guardsquare.com. (2016). ProGuard. https://www.guardsquare.com/proguard.

[17] Castellaro, M. y otros (2014). Threat Modeling: Designing for Security. En A. Shostack, Threat Modeling: Designing for Security (págs. 61-86). Wiley.

[18] Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Consultado el 07 de julio del 2013, en http://www.pmp-projects.org/Agile-Manifesto.pdf

[19] Davis, N. (2005). Secure Software Development Life Cycle Processes: A Technology Scouting Report (pp. 14–20).

[20] H. Mouratidis and P. Giorgini. "Integrating Security and Software Engineering". Idea Group Publishing. USA. 2007.