Mobile Software Security Threats in the Software Ecosystem, a Call to Arms

Andrey Krupskiy(✉), Remmelt Blessinga, Jelmer Scholte, and Slinger Jansen

Utrecht University, Utrecht, The Netherlands
{a.krupskiy,r.d.blessinga,j.scholte2}@students.uu.nl,
slinger@slingerjansen.nl

Abstract. This paper studies security policies of the Android and iOS software ecosystems. These platforms have experienced security issues since their public release in 2007. This research creates an overview of the results that security issues cause and the actions available to limit security infractions based on scientific literature. Following the overview, this paper attempts to explain premises of those issues by analyzing the security recommendations of both platforms and comparing them to OWASP security guidelines. This is done by comparing development guidelines set up by both platforms and assessing the importance of each of these guidelines in the ecosystem perspective. The conclusion highlights vulnerabilities in the developer guidelines of mobile platforms and recommends appropriate action to improve the situation.

Keywords: Software ecosystems · Software security · OWASP · Development policies

1 Introduction

As smartphones increased in popularity, so did the research in the field of smartphone security. Smartphones rely on the security architecture of the mobile platform that supports them. These platforms have different architectures, aside from sharing several issues each one is also subjected to security issues specifically harmful for the platform. This paper takes a look at the security issues of the mobile platforms Android and iOS which both have a marketplace called app stores. App stores are typical occurrences of a software ecosystem which is defined by Jansen et al. (2009) [1] as: a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them [1]. The marketplace opened up smartphones for third-party developers as is also recognizable in personal computer platforms [2]. As personal computer platforms have struggled with security, open marketplaces for mobile platforms resulted in a challenge for smartphone security.

Software created by third-party developers for mobile platforms are called smartphone applications or apps which, as a result of interacting with the platform, cause security issues. The OWASP foundation is a non-profit organization

A. Ojala et al. (Eds.): ICSOB 2017, LNBIP 304, pp. 161–175, 2017.
https://doi.org/10.1007/978-3-319-69191-6_11

which tries to bring visibility and evolution in the safety and security of the worlds software. Most relevant for this paper is the OWASP top ten list of security flaws that applications contain. These flaws adversely affect the actors in the software ecosystem and therefore mobile platforms should support developers in creating secure applications. This paper attempts to gain insight into the main security concerns of open mobile platforms and standard prevention methods. This paper will refer to OWASP as the golden standard of security which means that conclusions are based on how well results fit with the OWASP recommendations. Therefore the main research question is: To what extent do the developer guidelines for mobile platform application development follow OWASP security recommendations? Literature is consulted to create a general overview of the security concerns and some methods of prevention in the Android and iOS platforms. Next, how security issues affect the actors in the mobile platforms. Lastly, the Android and iOS guidelines are analysed in context of the OWASP top ten in order to see if they are an effective prevention method.

The structure of this paper is as follows: Sect. 2 is the research approach which describes what is done in order to research this topic. Section 3 contains literature overviews of the Android and iOS security. OWASP and the Android and iOS developer guidelines are researched in Sect. 4. Sections 5 and 6 contain the discussion and conclusion respectively.

2 Research Approach

To answer the main question, two research questions (RQ) are defined: RQ1: To what extent are ecosystem actors vulnerable to security threats? RQ2: Are there any significant security vulnerabilities in the developer documentation?

A literature overview creates the context for the RQ1 since it delves into the security issues regarding the mobile platforms. It approaches several issues of a mobile platform in an attempt to highlight the most prevalent. The method to collect literature regarding these topics consists of researching common security threats. Papers are consulted afterwards in order to create an detailed explanation and understanding of the threats and their possible solutions. References found in papers that are useful for this research are also researched. This creates a snowball effect resulting in detailed information about a topic.

To adjust from prevalent issues and to place the OWASP recommendations in a refined ecosystem perspective, actors are defined and an assessment is made to estimate to what measure ecosystem actors are vulnerable to security threats. The top ten list created by OWASP is explored and used to see if it addresses the security threats for the ecosystem. This allows to find specifically which issue in OWASP has an impact on which ecosystem actor and together with the literature overview answers RQ1. The issues addressed in the overview and derived from the ecosystem perspective can be compared to the findings of RQ2.

Lastly this research is focused on reviewing development guidelines of Android and iOS platforms in terms of security. Using the top ten list of OWASP mobile security recommendations the security policies of the Android and iOS

platforms are analysed. This results in a comparison of the two platforms and an overview of their security guidelines. The comparison results in a table with the complete overview offering an answer to RQ2. It is important to note that some guidelines have different terminology than used by OWASP, this is taken into account by searching for similar keywords and by looking at the underlying security concern instead of just the name of the guideline.

3 Literature Overview

This section contains the overviews derived from the literature studied on Android and iOS security in order to picture the circumstances in said field. The split between users and developers is based on Jansen et al. definition of an app store which has two distinct groups; users and developers [3]. In this section the Android and iOS developers are assumed as not wanting to benefit from deliberately releasing malicious applications or misusing user information. They are instead assumed as wanting to provide secure applications for the users, themselves and the software ecosystem as a whole.

3.1 Literature on Android Ecosystem Security

A literature survey by Rashidi and Fung [4] summarizes security threats caused by applications to a users' privacy and device. This survey consists of Android security threats and solutions collected from literature dating from 2010 till 2015. The three relevant threats are; 1: Information leakage, 2: Privilege escalation, and 3: Repackaging applications. Information leakage occurs since Android uses a permission-based system which applications use to gain access to, possibly sensitive, data on a phone [5]. A newly downloaded application adheres to this system since it has to explicitly request permissions up-front to access personal information and phone features [6]. In 2012 the permissions requested by 44% of the applications exceeded the minimum number of permissions needed to function. This violates the principle of least privilege, a longstanding principle in the world of computer security [7], it defines the importance of software only accessing the minimum information to function. Davi et al. [8] conclude from their research that the permission based system damages security for the user as it allows for privilege escalation attacks. Furthermore, Meng et al. [9] state that some OEM weaken the existing security of the device by customizing Android image that may lead to privilege escalation attacks or information leakage.

Android developers can opt to make earnings from their application by integrating an in-app billing service. Muliner et al. explain this service as follows: The in-app billing service allows users to pay for options, services, subscriptions, and virtual goods from within mobile apps themselves [10]. In their paper they developed an attack against the in-app billing service allowing them to bypass paying for in-app services in 60% of the 85 most popular applications. Another issue, one which can lose developers revenue streams and credibility is repackaging of their application Applications that normally cost money to use or have

an in-app billing service can be repackaged and then released for free on the Internet. Repackaged applications, like banking applications, can be distributed on the Android store.

Reverse-engineering of Android applications is impossible to prevent without violating the open policy of Android. There is however a tool provided by Android that defends applications from being simple reverse-engineering targets without violating the open policy. The Android provided tool Proguard changes features, like class and variable names, of a code to random strings. This technique is called code obfuscation, a way to prevent analysis of the code of an application. For the security issues that concern application users, developers have a bigger array of options to the threats mentioned earlier. Following Rashidi and Fung [4] developers can choose for one or two different types of strategies in regards to securing their applications. The first is making use of existing techniques and mechanisms which consists of using options provided in the Android OS. The second is to use software created by others which can detect security threats.

3.2 Literature on iOS Ecosystem Security

Academic literature regarding iOS security issues shows that user privacy can be at risk. Large scale mobile application analysis of iOS apps conducted by Orikogbo et al. shows that a large fraction of apps contain references and make connections to domains using the HTTP scheme [11]. Around 26% of apps studied just use HTTP connection schemes while 72% of apps use both HTTP and HTTPS and 2% use only HTTPS. Since personal information can be sent to a remote service, usage of HTTP is a privacy threat, since this kind of connection is not encrypted in any way and can be read by a third party if intercepted.

When examining the literature regarding security and the iOS system it is shown that there have been cases of security breaches in the iOS built-in security systems. A paper by Heider and El Khayari [12] shows that up to the iOS version 6.0.1 there was a security flaw that allowed to perform an attack on the iOS keychain service, which allowed to get an access to all data stored on the device and perform jailbreaking of the device. Jailbreaking is defined as a process of getting a root access to the device. Root access provides elevated privileges and allows user to avoid most restrictions of iOS enforced by Apple.

Additionally, Renard [13] describes a number of attacks that can be performed on iOS. First, Renard points out attacks that can be performed without jailbreaking a device. Those include getting access to the data of applications and the users credentials, retrieving data from a devices backups, monitoring communications, attacking secure communications to the server. In case of a jailbroken device it is possible to gain access to all data stored on the device, making reverse engineering an issue. For several versions of iOS it was possible to jailbreak a device without having an access to device's keychain and then hide the fact of a jailbreak.

The iOS platform makes apps work in a strictly restrictive environment [14] and it has two important security features: the app vetting process and the app

sandbox. Although these two features have proven to be effective for the iOS ecosystem, as shown in the survey conducted by Felt et al. [15] there was no harmful malware present in the iOS appstore, both methods still have some security issues, code signing can be evaded for example as discovered by Miller [16].

Another security issue that iOS developers encountered was an XcodeGhost attack [17]. This attack was designed so that app developers would download an infected copy of Xcode, a development environment for iOS apps, which would infect any apps created within it with malicious code. This is an example of the developers not following security protocols and using software downloaded from the not verified source. The paper by Renard [13] gives several suggestions on how developers can avoid these security threats. These suggestions include code obfuscation, creating kill-switches to delete user credentials in case of a reverse engineering attempt, secure usage of memory and some recommendations on how to prevent hooking. Hooking is as "a mechanism that allows users to alter or augment the behavior of applications".

A study by Teuf et al. [18] confirms, that jailbreaking a device allows to effectively disable the iOS file system encryption, allowing attacker to gain access to all data without knowing the passcode of the use.

4 Results

4.1 OWASP Guidelines and the Ecosystem Perspective

In this section each of the ten points is placed in an ecosystem perspective, enabling to take a closer look at each threat and estimating what actors in the mobile ecosystem are impacted by each vulnerability should they be left unprotected.

In their 2013 literature review Hansen and Manikas [19] identified five of the most common types of actors in software ecosystem literature. Each of these roles (Orchestrator, Component Developer, External Developer, Vendor and Customer) is directly affected by the security vulnerabilities mentioned in the OWASP top 10. In the context of this research we can identify these roles as the following ecosystem elements:

1. **Customer.** Also called end user this is the person. This actor is responsible for bringing monetary value into the ecosystem. If customers feel insecure, for example when their privacy is at risk, they might abandon an ecosystem or it might affect their purchasing behavior, affecting every involved actor.
2. **Orchestrator.** These are the ecosystem platform owners. Affected primarily in loss of reputation or competitive advantage when ecosystem security is threatened.
3. **Component Developer.** Also called the niche player, the app developers contribute directly to the ecosystem by providing its content applications.
4. **External Developer.** These are often passive participants in the ecosystem. They play a non-developing role in the ecosystem. These actors can

for example offer certain supportive services such as ad networks to component developers. They are mostly impacted by loss of clientele or reputation damage if the ecosystem performs badly.

5. **Vendor.** Also called the reseller or added value reseller. This could be for example a game publisher that buys game apps from component developers and sells them under the umbrella of cross-app marketing efforts. They are vulnerable to security threats due to being the face of the product, thus being held responsible for the product.

4.2 The OWASP Mobile Security Top 10

OWASP Mobile Security Top 10 in Context of SSN Roles. Looking at the top 10 mobile security threats defined by OWASP it becomes clear what each security threat entails and how to defend against such threats. Yet what actors are affected by such threats is not immediately clear, and neither is how the ecosystem as a whole is affected. The technical and business threats highlighted by OWASP do give us sufficient information to extrapolate what each threat means for the ecosystem entire. This section looks at how the security threats of the mobile top 10 can affect the software ecosystem, this is followed by a threat analysis for each ecosystem actor in Sect. 4.3.

1. **Weak Server Side Controls.** In the ecosystem perspective this means customer data can be lost or affected during an attack. It can also mean that an app becomes unusable.
2. **Insecure Data Storage.** In the ecosystem perspective this means that customer privacy and security is at risk from attacks. In extreme cases sensitive data of developers or vendors could also be at risk.
3. **Insufficient Transport Layer Protection.** In the ecosystem perspective this means that customer data is at risk from interception, this might not only affect customer privacy but also external developers that rely on the exclusivity of specific data.
4. **Unintended Data Leakage.** In the ecosystem perspective this means that customer privacy is at risk from attack, damaging the reputation of and trust in some ecosystem actors.
5. **Poor Authorization and Authentication.** In the ecosystem perspective this is primarily a risk for customers, but hacked accounts can be of concern to component developers or vendors who rely on the proper usage of their apps, the spread of spam through such accounts could for example damage its profitability. Malicious users might be able to obtain privileges they should not have.
6. **Broken Cryptography.** In the ecosystem perspective this affects customer privacy violations. It can also lead to information theft, code theft, intellectual property theft and reputation damage, affecting vendors.
7. **Client Side Injection.** Not only is this a risk to the customer due to privacy violations but the component developer as well as their security precautions could be directly affected by injected code. In extreme cases the orchestrator reputation might be affected.

8. **Security Decisions Via Untrusted Inputs.** In the ecosystem perspective this is a threat primarily for the actors that can be at risk from users with malicious intentions, customer data might be vulnerable to attack.
9. **Improper Session Handling.** Customer privacy and security could be at risk from attack. Component developers might be directly affected by security breaches, vendors might experience an interruption in common business procedures.
10. **Lack of Binary Protections.** In the ecosystem perspective a lack of binary protections is of great concern to all involved actors. A reversed engineered app is a great way for malicious parties to spread malicious content, relying on the trust customers place in certain ecosystem actors. Duplicate apps are a security risk. A hacked app can lead to privacy or confidential data theft. For the orchestrator it will lead to brand damage and revenue loss through pirating.

Table 1 shows an overview of the relations between ecosystem actors and how each actor is affected by the OWASP security top 10 threats.

4.3 Comparing the Effect of Security Threats to Ecosystem Actors

Given the five types of roles and how they are affected by security vulnerabilities, we can compare them to the OWASP mobile top 10. Table 2 shows the security threat identified by OWASP in the first column, followed by a threat assessment for each role of either low (no or little direct impact), moderate (some threats but either case specific or limited impact) or severe (certain to suffer some damage). The following table is primarily intended to give an overview and visualization of security threats to the ecosystem as a coherent set of actors.

The further away from using or directly developing the application with the possible security vulnerability, the less security risks affect the actor. Customers are shown to be most at risk from vulnerabilities in the ecosystem, followed by component developers while the orchestrator is shown to be least vulnerable.

4.4 Android and iOS Developer Guidelines

Both ecosystem orchestrators offer comprehensive guidelines for their developing partners. The Android guidelines are set up in a training and reference format, taking on the role of teaching the developer using any means deemed effective. Apple structures the documentation as a reference guide instead of guiding the developer through a series of trainings, as such this documentation is primarily text based and most effective when searching for specific advice.

In terms of security the iOS guidelines include a section named the Secure Coding Guide while Android offers a section under the header Best Practices for Security & Privacy. The Secure Coding Guide by Apple is the central reference point for their security recommendations, and they always link back to this document. Android is less inclined to discuss security concerns in other parts of the documentation, concentrating most advice in the best practices section.

Table 1. Security threats for the SSN stakeholders

Group	Activity	Needs	Cooperative / Financial Relationship					Security Threat
			Customer	Orchestrator	Component Dev.	External Dev.	Vendor	
Customer	Purchase product	Privacy, trust, exclusivity of data, reliable and valuable marketplace		X	X		X	Breach of confidentiality / privacy, loss or theft of sensitive data, loss of trust in the ecosystem, loss of ecosystem investment
Orchestrator	Support Ecosystem	Brand power, stable market environment, competitive advantage	X		X		X	Brand power damage, loss of ecosystem partners, loss of ecosystem power, loss of competitive advantage, possible decrease of revenue
Component Developer	Develop and deliver ecosystem content	Reliable open platform, brand power, reliable marketplace	X	X	X	X	X	Loss in revenue, brand power damage, loss of trust in ecosystem, loss of competitive advantage, possible decrease in ecosystem participation
External Developer	Develop and deliver supportive services	Stable market environment,			X		X	Loss in revenue, brand power damage
Vendor	Resell products	Brand power, stable market environment, open platform, good component dev. relations	X	X	X	X		Loss in revenue, brand power damage, loss of partnerships, loss of competitive advantage, possible decrease in ecosystem participation

4.5 Guidelines Comparison Table

Comprehensively studying the developer guidelines allows for the creation of a comparison table to see if and how Android and iOS make security recommendations suggested by OWASP in the mobile top 10. There are three levels designed to indicate if each guideline is present:

Insufficient: There is no advice or reference present in the developer guidelines on how to develop for the security vulnerability, developers are not made aware of the security risks. If a developer depends on sole advice offered by the guidelines this will result in an insecure application.

Table 2. Vulnerability effect on ecosystem roles. L is low threat, M is moderate threat, S is severe threat.

OWASP guideline	Customer	Orchestrator	Comp. Dev.	Ext. Dev.	Vendor
Weak server side controls	M	L	S	L	L
Insecure data storage	S	M	S	M	S
Insufficient transport layer protection	S	L	M	M	L
Unintended data leakage	S	M	S	L	M
Poor authorization and authentication	S	M	S	L	M
Broken cryptography	S	M	S	L	S
Client side injection	S	M	S	M	M
Security decisions via untrusted inputs	M	L	M	L	M
Improper sessions handling	S	L	M	M	M
Lack of binary protections	S	M	S	M	S

Partial: Some advice is given or the guideline is mentioned but not comprehensively enough to adequately assist developers to secure an application. Alternatively the security recommendation in the documentation is outdated.

Sufficient: Either coding guidelines, explanation of the security vulnerability with recommended precautions or specific advice on how to prevent the vulnerability is given. If a developer relies solely on the developer guidelines it will not pose a danger to application security.

When the results of the comparison table are transformed to scores (where insufficient is 0%, partial is 5%, and sufficient is 10%) the results show a 55% completeness score for iOS and 60% completeness score for Android. The Android documentation has six points out of ten that could do with improving, the iOS documentation has five points of out ten that could do with improving.

IOS Explained. The following overview explains where and how each OWASP guideline is present in the developer documentation (Table 3).

1. The Apple developer documentation focuses on how to handle authentication when exchanging information with a server. No mention of server configurations, backend services or best practices when setting up a server could be found.
2. Apple refers to storing information in the appropriate directory and setting the right file system permissions. Apples File Protection mechanism is considered to be safe for use for consumer-grade data. The documentation states that the various APIs should be sufficient. This is contrarian to the OWASP recommendation that developers should consider adding an additional layer of encryption.
3. Apple recommends choosing the appropriate transport protocol and highlights some concerns for each option. A set of secure networking pages is available, offering comprehensive guidelines and coding recommendations to create sufficient transport layer protection.

Table 3. Comparison of OWASP, iOS and Android developer guidelines.

OWASP guideline	iOS guidelines	Android guidelines
Weak server side controls	Insufficient	Insufficient
Insecure data storage	Partial	Sufficient
Insufficient transport layer protection	Sufficient	Sufficient
Unintended data leakage	Insufficient	Partial
Poor authorization and authentication	Sufficient	Sufficient
Broken cryptography	Sufficient	Sufficient
Client side injection	Sufficient	Partial
Security decisions via untrusted inputs	Sufficient	Partial
Improper sessions handling	Insufficient	Insufficient
Lack of binary protections	Insufficient	Partial

4. Apple does not refer directly to data leakage, nor to the ways mentioned by OWASP on how data leakage could occur on iOS. They consider the platform to be inherently secure due to apps being restricted in the files and system resources it can access.
5. Apple has a number of pages dedicated to authentication and authorization, they offer various coding recommendations as well as best practices and explanations on why or how something should be build to be considered secure.
6. iOS applications are, in theory, protected from reverse engineering via code encryption. Apple offers comprehensive explanations on cryptography topics, an API to use for cryptographic tasks and coding guidelines on how to securely implement cryptography.
7. Apple has dedicated an entire page to this security vulnerability, offering coding advice, examples of risks and information on injection attacks.
8. Comprehensive advice on how to validate input is present. Coding guidelines are offered, as is information on what kind of vulnerabilities might lead to security breaches and how.
9. The Apple developer guidelines offer no recommendations, coding advice or information on secure sessions handling.
10. Apple does not refer to the risks of not including binary protection and relies solely on its app review and submission process, binary encryption is central to iOS. However, this process is vulnerable to attacks when jailbreaking a device [13]. Apple does not recommend to developers that they take additional action such as jailbreak detection or certificate pinning controls.

Android Explained. In this section two main Android guidelines were used as a source: a training guide for developers and a guide for android source code.

1. The Android documentation does not provide any guidelines for setting up a server.

2. The Android guidelines describe all possible ways of storing app data on the device and secure ways of sharing data between apps. These guidelines also provide information on how to implement app data encryption and handle sensitive data.
3. The Android security guidelines dedicate a section to securely implementing HTTPS and SSL.
4. The Android guidelines mention the problem of data leakage and provide advice for some cases on how to avoid such a risk. There are some recommendations on how to work with log files, regarding them as being potentially vulnerable. The guidelines do not cover URL caching, keyboard press caching, Copy/Paste buffer caching, application backgrounding, HTML5 data storage, browser cookie objects or analytics sent to 3rd parties, which are mentioned in the OWASP guidelines.
5. The Android documentation has a number of sections describing authentication procedures. The documentation has a section dedicated to implementation of OAuth2 Services, which is an open standard for authorization.
6. Android provides a number of recommendations on how to implement cryptography. The guidelines encourage developers to use standard protocols instead of creating their own, this approach is also recommended by the OWASP guidelines.
7. Android guidelines acknowledge the danger and offer security advice of how to prevent XSS and SQL and JavaScript code injection on Android devices. Nevertheless, the list of security issues in Android guidelines does not include some problems mentioned in the OWASP security recommendations.
8. Android guidelines provide basic information on input validation methods, input validation security threats, and also state that Android has a number of countermeasures build in to prevent input related security problems. The guidelines do not provide concrete examples of tools used to reduce this security threat. Furthermore, no coding examples or best practices are described in this section.
9. Android security guidelines provide no information on secure session handling.
10. Android provides some guidelines on binary protection in context of Google Play in app billing. These guidelines suggest signature verification, code obfuscation and modifying sample code for in app billing system to decrease the ease of its detectability. OWASP also mentions a root detection problem. Rooting an Android phone is similar to jailbreaking iPhone, but Android guidelines provide no information about security in context of rooting.

5 Discussion

Security is one of the big issues for developers who have the optimal security for themselves and their users in mind. In this paper the belief is that developers want the most secure software ecosystem as to benefit the actors that participate. Therefore something not touched on is that there are also developers and organizations who intentionally force privacy risks on users.

This research does not include the orchestrator actors perspective on the importance of ecosystem security aspects, interviews with Apple or Google would have been an excellent source of data but setting this up did not fit in the scope of this research. Additional research could be done to investigate why both the studied platforms did not include specific OWASP guidelines and the role of the developer guidelines in the development process. One way to execute such research is to create app security conceptualization similar to mobile application usability conceptualisation performed by Hoehle and Venkatesh [20]. For example, the most recent research on iOS apps, which included an inspection of almost 42.000 apps, showed that almost 26% of apps reference external resources strictly via HTTP, which is considered to be an insecure way of transferring information by OWASP, iOS and Android guidelines. Both iOS and Android guidelines actively encourage developers to use the HTTPS protocol, which is considered to be far more secure, and provide detailed guidelines on how to implement it.

Although attacking a jailbroken iOS system was proven to be a much simpler task for attackers, no literature or reports were found regarding a working mechanism for jailbreaking a device running iOS 10 protected with passcode without knowing a passcode for the device. This leads to conclusion that at this time, passcode of sufficient length and base (number of characters used to create a passcode) serves as a sufficient way of protecting a device against jailbreaking, this does not take into account for the risk of social engineering being used to recover the passcode or user negligence when setting a passcode.

It should be mentioned that the mobile top 10 list dated 2012 was used. It was considered to use mobile top 10 dated 2016, but this list is still in development and incomplete.

This paper was written from the perspective of security in the software ecosystem domain. As such it did not look at the quality of advice and information offered in the developer documentation. This is considered a task more suited for security experts.

6 Conclusions

The literature used for the overview shows that there are several issues apparent for the two groups that participate in app store based mobile platforms. For user issues both ecosystems are lacking in offering full protection. Both ecosystems have different foundational functionalities that cause insecurities for the users. Developers seem to encounter more problems on the Android platform, mainly revenue based, in comparison to iOS developers since the iOS environment is more restricted. Both platforms however also provide integrated and external options for developers to create secure applications for both themselves and the users. These options do not always offer a working solution in regards to one of the issues.

Justified by the results from the actor/OWASP guideline impact evaluation on a software ecosystem level, it is concluded that the biggest impact of security breaches is felt by customers and component developers. This aligns with the

literature overview which reveals that there are numeral issues affecting these groups. Vendors are moderately at risk, depending on how close they are to the direct development, management or publishing of the application. External developers and the ecosystem orchestrators have the least risk as they are furthest removed from the security vulnerabilities. This answers RQ1: *To what extend are ecosystem actors vulnerable to security threats?*

From the developer guideline evaluation results, it is concluded that the evaluated guidelines form a solid basis for the development of a secure application but can still be improved. The documentation for iOS offers a comprehensive security guide that helps with many issues not included in the OWASP mobile top 10. However, four out of ten points in the OWASP mobile top 10 are not sufficiently presented in these guidelines and one point offers incomplete advice, leading to the conclusion that the iOS guidelines need improving before they can be considered fully secure. The results of this study confirm that inherent protections can sometimes be circumvented, leading to the conclusion that additional advice should be offered to developers in case this occurs.

It can be concluded that the Android guidelines leave the impression of being a good starting point for the developer. However, the results show that some sections only acknowledge a security issue and let a developer either find a solution himself, or suggestively use a solution built into the Android framework, which has its downsides according to OWASP. The results from the comparison of the OWASP guidelines with Android guidelines allow for the conclusion that the Android guidelines should be further developed, particularly in a sense of improving existing sections with concrete solutions and best practices on how to deal with security threats.

It is difficult to decisively conclude if one of the platforms does a better job offering secure guidelines following the OWASP framework, as both have their individual strengths and weaknesses.

The results show significant security risks are posed by the incompleteness of advice on server side controls, secure data storage, unintended data leakage, client side injection, security decisions via untrusted inputs, improper session handling and lack of binary protections. This answers RQ2: *Are there any significant security vulnerabilities in the developer documentation?* with a yes. The significant risks as a result from incomplete advice correspond with the findings of RQ1 in regards to the issues users, developers and other software ecosystem actors encounter. This correspondence can be derived from the descriptions of the lacking guidelines and their interaction with the issues.

The comparison tables and subsequent evaluation of the results lead to the conclusion that neither platform adequately adheres to the security guidelines set by the OWASP mobile security project. This provides an answer to the main research question posed in this paper.

Final Conclusions and Recommendations. The literature regarding the security issues in both analysed ecosystems show that they are not completely secure for their users and developers. Customers do not always understand how

the applications can seriously affect their security. Developers are not always capable of securing their applications as a result of problems like repackaging. Problems like these add to the importance of secure ecosystems especially in the form of well-defined security guidelines that follow security recommendations.

The impact on the entire ecosystem was assesed when the OWASP framework was placed in the context of software ecosystems, it is considered of high importance to the security and health of the mobile software platforms that security guidelines are fully included in the developer documentation. Based on the conclusions presented in this paper, the recommendation can be made that platforms should consider including more comprehensive information on secure development using a framework such as OWASP. Regardless if the operating system has been designed with protections in mind, orchestrators should still include information on secure development in their documentation, as it has been shown that these measures can be circumvented in some cases and the additional measures taken by developers can only benefit the ecosystem. Some platforms already show the value of the OWASP guidelines by referring to them on the introductory page of the security guide, while others do no such thing. It is recommended that the value of these guidelines is more clearly referred to in the developer documentation.

References

1. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: a research agenda for software ecosystems. In: 31st International Conference on Software Engineering-Companion Volume. ICSE-Companion 2009, pp. 187–190 (2009)
2. Asokan, N., Davi, L., Dmitrienko, A., Heuser, S., Kostiainen, K., Reshetova, E., Sadeghi, A.R.: Mobile Platform Security Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers (2013)
3. Jansen, S., Bloemendal, E.: Defining app stores: the role of curated marketplaces in software ecosystems. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 195–206. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39336-5_19
4. Rashidi, B., Fung, C.: A survey of android security threats and defenses. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. (JoWUA) **6**(3), 3–35 (2015)
5. Wei, X., Gomez, L., Neamtiu, I., Faloutsos, M.: Permission evolution in the android ecosystem. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 31–40. ACM (2012)
6. Grace, M.C., Zhou, Y., Wang, Z., Jiang, X.: Systematic detection of capability leaks in stock android smartphones. In: NDSS, vol. 14, p. 19 (2012)
7. Saltzer, J.H., Schroeder, M.D.: The protection of information in computer systems. Proc. IEEE **63**(9), 1278–1308 (1975)
8. Davi, L., Dmitrienko, A., Sadeghi, A.-R., Winandy, M.: Privilege escalation attacks on android. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 346–360. Springer, Heidelberg (2011). doi:10.1007/978-3-642-18178-8_30
9. Meng, X., Song, C., Ji, Y., Shih, M.-W., Kangjie, L., Zheng, C., Duan, R., Jang, Y., Lee, B., Qian, C., et al.: Toward engineering a secure android ecosystem: a survey of existing techniques. ACM Comput. Surv. (CSUR) **49**(2), 38 (2016)

10. Mulliner, C., Robertson, W., Kirda, E.: VirtualSwindle: an automated attack against in-app billing on android. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 459–470. ACM (2014)
11. Orikogbo, D., Büchler, M., Egele, M.: CRiOS: toward large-scale iOS application analysis. In: Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 33–42. ACM (2016)
12. Heider, J., El Khayari, E.: iOS keychain weakness FAQ. Frauenhofer Institute for Secure Information Technology (SIT) (2012)
13. Renard, M.: Practical iOS apps hacking. GreHack 2012. 14 (2012). https://papers.put.as/papers/ios/2012/GreHack-2012-paper-Mathieu_Renard_-_Practical_iOS_Apps_hacking.pdf
14. Han, J., Yan, Q., Gao, D., Zhou, J., Deng, R.H.: Comparing mobile privacy protection through cross-platform applications (2013)
15. Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D.: A survey of mobile malware in the wild. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 3–14. ACM (2011)
16. Miller, C.: Inside iOS code signing. In: Symposium on Security for Asia Network (SyScan) (2011)
17. Meng, W., Luo, X., Furnell, S., Zhou, J.: Protecting mobile networks and devices: challenges and solutions (2016)
18. Teufl, P., Zefferer, T., Stromberger, C., Hechenblaikner, C.: iOS encryption systems: Deploying iOS devices in security-critical environments. In: 2013 International Conference on Security and Cryptography (SECRYPT), pp. 1–13. IEEE (2013)
19. Manikas, K., Hansen, K.M.: Software ecosystems-a systematic literature review. J. Syst. Softw. **86**(5), 1294–1306 (2013)
20. Hoehle, H., Venkatesh, V.: Mobile application usability: conceptualization and instrument development. MIS Q. **39**(2), 435–472 (2015)