Global Information Assurance Certification Paper

Technical and Non-Technical Perspectives for Auditing Mobile Applications

GIAC (GCNA) Gold Certification

Author: Muzamil Riffat, muzamil@hotmail.com

Advisor: Manuel Humberto Santander Pelaez
Accepted: TBD

Abstract

The latest research by leading institutions has demonstrated that the mobile phone users overwhelmingly prefer to use the mobile applications rather than the browser(s) installed on their devices to access the digital content. With the ever- increasing market penetration of smartphones, the usage of mobile applications is anticipated to increase even further. In addition, as more devices connect to the Internet due to the advent of "Internet of Things", it is anticipated that even more mobile applications will be developed to control and monitor these devices. Like so many other trends in technology advancement, the security of mobile applications and the associated risks have not been given the due consideration. This paper presents a framework and methodology for auditing mobile applications that can be used by the technology auditors or the security professionals to comprehensively audit and assess the security posture of the mobile applications, powering the business of their organizations. The paper not only discusses different technical approaches for the assessment of mobile applications, but it also focuses on other areas such as the functional and nonfunctional considerations, market pressure to rapidly release the specific mobile applications and the intellectual property rights related to the mobile applications.

1. Introduction

Technological advancements have impacted all aspects of our life. In the field of telecommunication, these advances can be witnessed in the form of increased usage of electronic mobile devices. Modern mobile devices offer several features such as calling, texting, the ability to carry out financial transactions, e-mails, etc. These devices offer "access to the Internet through interfaces such as Bluetooth, WLAN, infrared or GPRS, TCP/IP protocol stack; desktop PC synchronization; the ability to simultaneously run multiple applications; open Application Programming Interface (APIs) to develop the applications" (Armin, 2013). As shown in Accenture's Electronics Products and Services Usage survey (2013), the consumers are increasingly expecting that the mobile devices should provide all features available in a traditional computer. Even at the workplace, more and more organizations are allowing employees to connect their devices to the organizational network. Aberdeen Group's recent study found that 75 percent of the participating organizations allowed their employees to utilize their own personal mobile devices for work-related communications (Zielinski, 2012). Further research in the usage pattern of the mobile devices indicates that mobile phone users prefer to use the mobile applications rather than the browser(s) installed on their devices.

With the ever- increasing market penetration of smart-phones, the usage of mobile applications is anticipated to increase further. In addition, as more devices connect to the Internet and the so-called phenomenon of "Internet of Things" grows, it is anticipated that even more mobile applications would be developed to control and monitor these devices (Mobilefuture, 2017). According to the research carried out by Nielsen, consumers spend approximately 90% of the mobile media usage time with the apps, and 10% with the mobile web. The mobile devices and the applications installed on these devices are an attractive target for cybercriminals because the devices "are always connected, they always carry some personal data, and they are even equipped with small cameras, microphones, and positioning devices" (Muttik, 2011). With the increased usage of mobile devices, information security concerns also need to be considered.

Application-level threats appear to be most widely discussed in the literature (Faruki et al., 2015). Since mobile devices can execute apps, the malicious actors

typically use the apps as the target vector to compromise the security or information contained in the mobile devices. For example, researchers demonstrated the possibility of extracting data from mobile devices using inaudible sound waves (Do et al., 2015). Enough attention is not paid to security details as the apps are typically developed with the focus on functionality (D'Orazio and Choo, 2016). For example, the research by D'Orazio and Choo (2015) noted several vulnerabilities related to data privacy in an app used by Australian government healthcare department. Farnden et al. (2015) also showed similar vulnerabilities in nine popular dating apps. It is, therefore, important that a comprehensive auditing and testing methodology be used to mitigate concerns related to mobile application development and deployment.

The main purpose of initiating mobile auditing or testing program is to "identify information security vulnerabilities of the application that could be exploited by a nefarious actor" (AppSec Labs, 2014). Along with assessing the functionality of the apps, the security assessment is vital to "mitigate risks and avoid devastating data breaches. A data breach can have a deep impact not only on the immediate bottom line; but it can also undermine customer confidence and loyalty, and damage your brand reputation" (Hayes & Rangarajan, 2012). The data security is about internal education, and consistent and dynamic security practices to continue auditing the infrastructure around who has access to what information (Walberg, 2013). If the appropriate security controls are not implemented, and the app is vulnerable to the security exploits, an unauthorized individual can have access to all the data present on the device, including passwords, user identification credentials, contact lists, banking information, other apps, and miscellaneous data stored on the device (Enderle, 2012).

This paper presents background knowledge that can be used by the technology auditors or the security professionals to comprehensively audit and assess the security posture of the mobile applications, powering the business of their organizations.

2. Growth of Mobile Devices Usage and Breaches

Due to the high cost of ownership, the mobile devices were initially owned only by financially privileged members of the society. However, the dramatic decrease in the

cost of the hardware along with the other technological developments such as more computer processing power at a lower cost has allowed the access to mobile devices to all sections of the society. For instance, Pew Research Center Technology Ownership Survey in 2015 showed that American adult ownership of a smartphone had increased to 68% in 2015 from 35% in 2011. According to the American global media measurement and analytics company "comScore", the number of mobile-only adult internet users in the USA exceeded the number of desktop-only internet users in March 2015, as indicated in **Figure 1: Single Platform Users' Share** below:

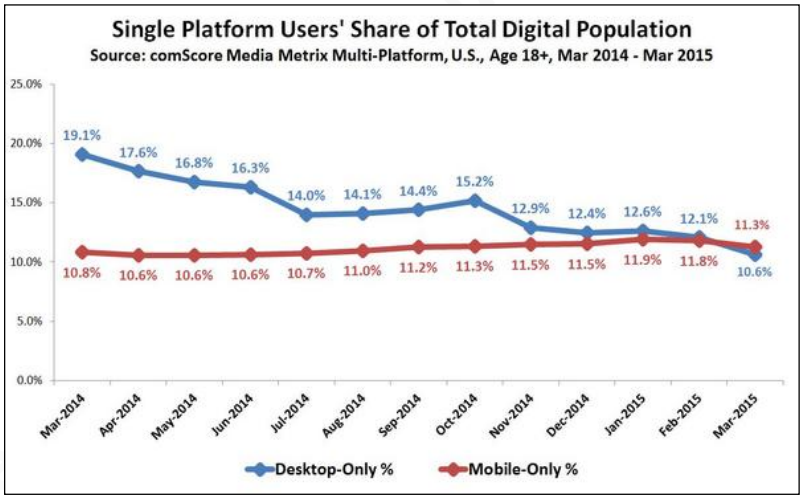


Figure 1: Single Platform Users' Share

Source:
https://www.comscore.com/fre/Insights/Blog/Number-of-Mobile-Only-Internet-Users-Now-Exceeds-Desktop-Only-in-the-U.S

As of 2014, global mobile users are more than the global desktop users as shown in **Figure 2: Number of Global Users** below.

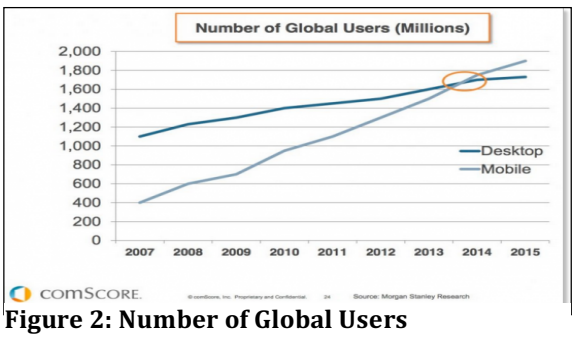


Figure 2: Number of Global Users

Muzamil Riffat, muzamil@hotmail.com

Source: https://www.comscore.com

The survey conducted by Statistic Brain Research Institute in 2015 revealed the following information:

Statistics on Mobile Device Users	**Data**
Percent who said their mobile phone is the first and last thing they look at each day	29 %
Percent who said they need to have the latest mobile technology	18 %
Percent who check their mobile device every 30 minutes or less	37 %
Percent who said they could only go a few hours without their mobile phone	34 %
Percent who said they prefer to communicate by text message	32 %
Percent who have asked someone on a date via text	20 %
Percent who say their mobile device make them better parents	65 %
Percent who would take their mobile device to work over their lunch	66 %
Opinion on being constantly connected by technology	**Data**
Mobile technology is mostly helpful	76 %
Mobile technology is a burden	13 %
Don't know	11 %
Percent who do the following on their phone at least twice a week	**Percent**
Browse the internet	46 %
Listen to music	32 %
Search the internet	41 %
Read news or current events	36 %
Take pictures	36 %
Visit social networks	37 %
Play games	32 %
Navigate using GPS	18 %
Shop online	12 %
Receive payments	5 %

Muzamil Riffat, muzamil@hotmail.com

Video chat	9 %

Source: http://www.statisticbrain.com/mobile-device-cell-phone-statistics/

The information presented in the above Figures shows that the ownership and usage of mobile devices is increasing at a great pace. However, the rapid adoption and proliferation of mobile devices are accompanied by information security concerns. There are several research studies that indicate that mobile devices are increasingly used to compromise the data security. For instance, the study by Ponemon Institute[1] reveals 67% respondents out of 588 Global 2000 companies cite insecure mobile devices as a reason for potential data breach in their organizations. According to the same study, an infected mobile device can cost an organization approximately $9,485. Similarly, a study by Mobile Iron indicates that over 50% of the organizations have at least one insecure mobile device. These studies show only a snapshot of information security concerns with mobile devices, but they provide a realistic indicator of the trends in the realm of mobile device security.

3. Mobile Applications Landscape

This section explains the different components of the landscape in which applications are deployed and tested.

3.1. Types of the Apps

Mobile apps are broadly categorized into three types: "Native," "Hybrid" and "Web" (Knott, 2015). Native apps are developed for a specific mobile platform. Therefore, they are developed in the programming language supported by that platform. The native app for Android platform is typically developed in Java programming language, whereas iOS apps are developed using Objective-C or Swift. Native apps have full access to platform-specific software libraries and hardware resources. This access allows the native apps to have significantly higher performance than Hybrid or Web apps. Since the Native apps are optimized for a specific platform, they have very interactive user-interfaces. These apps are distributed through the “app store” of different service

[1] https://www.lookout.com/enterprise-mobile-risk

providers. The approval process of some providers might take a long time. The investment to develop the Native apps is quite higher as the separate code is developed for each platform.

Hybrid apps combine web technologies such as HTML and JavaScript for developing the client-accessible Web part, and the native mobile code that is specific to the platform. To compile the Web code into native mobile code, a hybrid development framework such as "PhoneGap"[2] is used. The hybrid development framework builds a bridge that allows the communication between the native code and the Web part. Since the Web part is a separate component, it can be independently hosted on a server. If minor changes are required, the app can be easily updated without going through the store approval process. The major drawback of the hybrid apps is that the Internet connection is always required to access the Web part. Furthermore, the performance might also be impacted since the Web part is accessed from the server.

An app that can be accessed using the Web browser of the device is called a Web app. These apps are independent of the mobile platform. Most Web apps are developed using HTML5, JavaScript and CSS technologies with a web interface supporting the native application look and feel (Knott, 2015). The use of these commonly technologies makes the development process of the web apps quite an easy task. The development and release process is further facilitated by the fact that no app store approval is required. Web apps have very limited access to the hardware of the device. Therefore, the functionalities offered by the Web apps are primitive. Although Web apps are mobile platform independent, the look and feel can vary depending upon the use of different browsers.

3.2. Business Models of the Apps

There are different business models used to generate revenues after developing an app. The most widely used method is the “freemium”(Apple, 2017) approach. In this model, a free app with core and limited features is made available to all users. To access the full functionality of the app, the user is required to make a payment. Advertisements

[2] http://phonegap.com/

can be displayed within the app to generate revenues. There are several advertisement frameworks such as the widely used "AdMob"[3] or “Search Ads”[4] which developers can use to implement advertisement features.

In the “paid” model (FbombMedia,2017) , the user must make a payment first before downloading and using the app. The paid model is typically used for gaming apps. When the user pays after completing a transaction, the app uses the "transaction model." PayPal is an example of transaction model whereas the fee is paid after sending or receiving the money. The research by Gartner (2013)_ has indicated that the frequency of download of paid apps is much less than the apps using freemium or transactional model. Therefore, the business model of the app should be carefully considered.

3.3. Architectures of Mobile Devices

It is essential for the app tester to obtain an understanding of architectures of mobile devices in order to make the testing process effective and relevant. The two leading architectures of mobile devices are Android and iOS. This section provides a brief summary of each. The pictorial representation of Android and iOS architecture is in Appendix A.

Android architecture is Linux-based, and it was developed by a major contribution from Google. "Linux kernel" is at the bottom of the architecture stack and is used to support functionalities such as scheduling of the processes, and memory and device management. It also has access to other hardware features including the drivers of the device. Any changes in the hardware orientation, e.g. screen rotation, are converted to software instructions by the device drivers, and these instructions are communicated to Linux kernel. Linux kernel contains all “drivers” for maintaining communication with the hardware elements of the device. The runtime layer of Android architecture is called “Android runtime” (ART). As of Lollipop version of Android, ART is the only runtime layer available. It allows for Ahead-of-time (AOT) compilation and improved garbage collection. To interact with the low-level components, “native libraries” are installed using the Native Development Kit (NDK). The “application framework” is used to

[3] https://www.google.com/admob/
[4] http://searchads.apple.com/

manage the development lifecycle of the application. It contains key services such as "Activity Manager," "Content Providers" and "Telephony Manager" for developers to build complex application functionalities. The "Application Layer" is the top most layer in the stack, and this is where the user interacts with the applications.

iOS architecture is used to run all Apple devices. iOS is Apple's proprietary architecture, and it can only be run on Apple devices. The iOS architecture is made up of four layers. Each layer is equipped with several frameworks that can be used for development purposes. The first layer is called "Cocoa Touch" and it contains key frameworks, and provides basic infrastructure for an app. The "Media" layer provides the audio-visual capabilities. The "Core Services" layer contains the vital services required for application development. Finally, the "Core OS" consists of low-level services such as OS X kernel and interacts with the hardware of the device. To assist the developers for building iOS apps, Apple has released an integrated development environment called "Xcode" (https://developer.apple.com/xcode/). Since 2015, Apple requires all applications to be developed using "Swift" programming language. iOS architecture allows for multiple layers of security. At the device level, security features such as passcode are enforced. At the system level, "secure boot chain" and "Secure Enclave" are used. Secure boot chain ensures that Apple digitally signs every step of the booting process, thereby preventing kernel -level attacks at device start up.

Secure Enclave is an independent process that monitors the device access. In the latest version of Apple processors, the unique keys used by Secure Enclave are not even known to Apple. The situation where even Apple cannot have access to the device has implications in the legal cases where Apple might be ordered by the court to provide access to the devices as it happened in a recent high-profile case where the court ordered to provide access to a terrorist's device. Encryption techniques through Data Protection API (DPAPI) are applied to provide data level security. Furthermore, each app is isolated from each other. Strict restrictions are placed on apps viewing the data or business logic of each other. Any required information is managed through permissions that need to be explicitly granted.

The process of removing restrictions imposed by the operating systems is known as “rooting” in Android, and "jailbreaking" in iOS. There are a variety of tools available that can assist in removing the restrictions. Rooting or jailbreaking allows “phone’s owner to gain full access to the root of the operating system and access all the features” (Siciliano, 2012, para.2). Jailbroken or rooted system "cannot put trust in its kernel [part of the operating system that controls computer hardware]. The modified kernel might disable security measures, contain malware such as key-loggers, or subtly alter the system's behavior to leak private information" (Lange, Liebergeld, Warg, & Peter, 2011, p.4). Although “rooting” or “jailbreaking” can be useful to auditors or security professionals in certain scenarios of testing an app, it should be kept in mind that this process might void a device’s warranty or support agreement. Therefore, personal devices should not be used for the testing purposes, and appropriate approvals should be obtained before carrying out rooting or jailbreaking.

4. Auditing Methodology, Tools, and Techniques

For some individuals in both general public and technology field, mobile apps are just a piece of software. The software testing process has achieved a certain degree of maturity as there are numerous Therefore, the techniques applicable to software testing could also be applied to app testing. There are, however, key challenges and differences that should be kept in mind during mobile app testing.

4.1. Challenges in Testing the Mobile Apps

The foremost challenge in testing an app is to perform procedures for testing the level of personalized attention expected by the user of the app. The study reveals that developers spend a little time in examining and considering the applications in term of its potential users to gain an understanding of what pleasurable experience the potential user expects (Boivie, Aborg, Persson, & Lofberg, 2003). The research conducted through several surveys has revealed that the expectations of the mobile app users are much higher than the usage of other software (Compuware, 2012) . The research also shows that an average user checks the mobile devices every six minutes, and the device contains the data that is highly personal in nature. Therefore, the expectations from the users are

that the app would provide a highly personal experience as well. Users tend to be very fickle in nature, and it is estimated that almost 80% delete an app after using it for the first time (Knott, 2015). One way of addressing the challenge of testing mobile apps for users' expectations is to create profiles of different potential customers with attributes such as gender, age, educational background, mobile usage habits, etc. and testing is performed from the perspective of those profiles. Services such as Mobile Personas[5] can be used to obtain general understanding about the behavior of different types of users. The concept of personas is a common way to assess the expectations, habits and other traits of the customers.

The testing for availability of a reliable data network poses another challenge. As the name suggests, the mobile user is quite mobile. Therefore, the testing needs to be done considering various network situations and weather conditions impacting network availability.

The app development or update cycle might not be able to cope up with the rapid releases and updates of the mobile devices. The history of the new releases from the leading phone manufacturers indicate that the updated model is released on an annual basis. According to the research by Open Signal, there were more than 20 thousand distinct Android devices in 2015. It would be a laborious task to test the app on all the devices. Similar to creating different groups for customers, different mobile device groups could be set up for testing the app. Particular attention should be paid to testing the app on the latest leading devices with the largest market share. For these devices, the testing cycle needs to incorporate the changes in the device features and processing power to ensure that the app can function appropriately both on the old and new phones. It is also important to monitor the market for the release of the new phone or the new features of the operating systems. The testing can also be done using Open Device Labs[6]. These labs provide free devices for app testing purposes. The devices are donated by the individuals or organizations to support such efforts. The availability of different kinds of sensors in mobile devices also needs to be tested thoroughly, and in various situations.

[5] http://www.mobilepersonas.com/

[6] http://opendevicelab.com/

For instance, an ambient light sensor adjusts the brightness of the screen depending upon the intensity of the light in different situations. Therefore, the testing should incorporate different test cases i.e. dark room, semi-lit room, outdoors, etc. Similarly, extensive test cases should also be documented for other sensors like proximity sensors, acceleration and gyroscope sensors (used to detect device's portrait or landscape orientation), magnetic sensors (mostly used to provide navigation-related information), environmental- related sensors (used for providing pressure, temperature, and humidity related information) and other communication hardware and sensors (microphones, camera etc.)

4.2. App Testing Roadmap

It is vital to document a roadmap to guide the app testing process. The roadmap document contains the testing approach as well as the required, basic information that establishes a common understanding of the objectives, tools, and techniques used for testing, the resources requirements, the success or failure criteria, and the conditions under which the testing procedure would stop. The document contains information about the specific features of the app so that those features can be thoroughly tested. Furthermore, the matrix containing the app features and the target demographic group should also be created, as the app usage habits of different demographic users might vary significantly. The most important element in the roadmap document is to specify the scope of testing. It is practically impossible to perform the testing in all different hardware and mobile platforms available. Therefore, the approach adopted to optimize the scope coverage is documented to establish a common understanding between all stakeholders.

Multiple approaches can be used to optimize the scope. In the "single-device" approach, only one device is selected for testing app functionality. The single-device method is typically used when there is only one device to be supported by the app, or if there is an immense release-to-market pressure. Since the testing is done only on one device, there is a risk that a lot of bugs or issues might remain unidentified during the testing phase. The second approach is called "multi-device." In this method, the testing is performed on selected multiple devices with the same or different mobile platforms. The

market-share of the devices on different mobile platforms could be used to select the devices. Furthermore, the information obtained from mobile usage tools such as "Our Mobile Planet"[7] could be used to select the appropriate devices based on the target user base. The customers and their expectations are one of the most important challenges of mobile application development and testing process. In order to satisfy the customers, it is vital that their needs are clearly understood. To achieve this, detailed information about customers' target group and their technological preferences should be obtained. The statistical information Internet pages, such as "Our Mobile Planet," maintain information based on geographical region, age, gender, user expectations and the kinds of devices users keep. The information about customers gleaned from different statistical sources is used to select the appropriate devices for testing. The "maximum device" approach attempts to perform the testing on as many devices as possible. This method is suited for the apps that have a reasonable degree of certainty of attracting the mass target audience. A significant amount of effort and research is carried out to identify all possible hardware and software combinations for testing purposes. The main criteria used to identify the devices are the geographic region and the demographic attributes of the target user group. If the app is developed with primarily focus on North American users, then the research is carried out to identify the most mobile devices used within North America region. Similarly, the research for app developed for the East Asian market will identify the most used devices in that region. Finally, the "use-case" approach dictates picking up just the important features of the app that impact the core business requirements for testing and leaving out insignificant functionalities. In this approach, the features related to the core business of the app are prioritized for testing. For example, if the app is developed for ordering food online, the use-case approach will prioritize testing that the food order can be placed and that the related financial transactions can be processed. The testing of other features such as the "Help" functionality will not be initially performed. If the time is still available at the end of the testing phase, the excluded features may be included in testing.

There are various Software Development Life Cycle (SDLC) methodologies that might be used to design, develop and test the mobile application. The most popular

[7] https://www.thinkwithgoogle.com/cee/planning-tool/our-mobile-planet-tool/

models of SDLC are: Waterfall, Iterative, Spiral and Agile models. A typical SDLC cycle goes through the stages of "planning and requirement analysis", "defining the requirements", "designing the product architecture", "development", "testing" and "deployment". The roadmap document also includes the description of testing that would be performed at different stages of the development lifecycle. For instance, the testing pass criteria at the unit-level testing of the "development" stage should be clearly documented. Similarly, there might be cases where UAT ("user acceptance testing") would be a mandatory requirement. Furthermore, alpha or beta testing to obtain potential customer opinion might also be included.

The information regarding the test data is an important element of the testing roadmap document. The test data format should be as close to the expected, realistic data as possible. An app typically processes three kinds of data including: the "configuration data" which includes the setup and backend operational information, the "stable data" which includes the information that is not expected to be frequently changed, and may include users' first or last name, e-mail, etc., and the temporary data, as the name suggests, which changes quite frequently. The temporary data includes data about the session management, cookies used during the information exchange, payment information or voucher codes. If possible, the test data should be saved for future testing. Storing the test data in a database may provide the possibility of automating the testing in future.

To monitor the performance of the app, the crash reports contain valuable information regarding the problems in the app. The app testing team is generally interested in obtaining information about the app crashes by criticality ranking, grouping and categorizing the potential problem. There are many tools such as the widely-used "HockeyApp"[8] or "Crashlytics"[9] that can assist in managing the crash reports. These tools also have the capability to display graphs showing app crashes over a defined period of time. Some app stores also provide basic crash information.

[8] http://hockeyapp.net/features/

[9] http://try.crashlytics.com/

App testing can be performed in simulator, emulator or a real-device environment. The simulators, such as the iOS simulator[10], attempt to simulate iOS devices with current and legacy operating systems. The simulators are used for testing iOS apps. On the other hand, the Android emulators[11] create a desktop environment with device's architecture. Although the testing related to sensors might not be carried out in a simulator or an emulator environment, both techniques are valuable for performing testing in the initial stages of the development. Since the app testing should also be done in a real life mobile environment, the roadmap document should include details regarding the extent of usage of a simulator or emulator and the physical device testing.

The app testing can be performed in a manual or automated manner, or through combination of both methods. The automated testing is generally performed to test the flow of the business processes in a controlled testing laboratory environment. The manual testing, on the other hand, is typically used to test the application behavior in real-life physical environments e.g. while walking, in a train or in different weather conditions. The degree to which manual or automated testing would be employed should be clearly mentioned in the roadmap document. One approach is to first perform adequate manual testing on different devices, and then to perform automated testing to test different features of the application extensively. Relying on only one method, manual or automated, might not be sufficient because there are certain features, such as temperature testing, which can only be assessed during manual testing.

4.3. Auditing Methodology and Threat Modeling

If an app testing project is considered a journey, then the road map document serves as the navigation document. The threat modelling of the application defines how the testing will be carried out. While auditing an app, it is essential to use a comprehensive methodology to ensure that all aspects of the app will be adequately audited. This section provides a brief introduction to some of the threat modeling methodologies that can be used.

[10] https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide

[11] http://developer.android.com/tools/help/emulator.html

Microsoft developed a threat classification method called "STRIDE." The first letter in the acronym represents "Spoofing," and refers to the ability of the users to assume the characteristics of another user. Manipulating the session ids, social engineering, and other malicious apps using the session id are a few examples of spoofing the identity.

"T" refers to "Tampering," which is the process of impacting the integrity of transactions carried out by the app. The tampering of data can be performed if a malicious user has access to the device. The tampering can also occur during GET and POST transactions, or during a manipulation of cookies or HTTP headers. A malicious user can also set-up a fake network access point, or have unauthorized access to a network access point to tamper the data.

The letter "R" represents "Repudiation". Without adequate logging and tracking, a user might deny or dispute the transactions. The app should be thoroughly assessed if enough controls have been implemented for non-repudiation. The assessment of non-repudiation is specifically necessary if the app handles financial or other sensitive data.

The next category is "Information Disclosure," which refers to the inappropriate disclosure of private information. An app might store sensitive information in an unencrypted format. Therefore, if the device is lost or some other malware compromises the security of the device, the confidential information might be disclosed. While performing the assessment, it should be assessed that the sensitive information contained within the app has enough protection controls from inadvertent information disclosure. The apps can also be subject to "Denial of Service (DoS)" attacks. The DoS attacks can be carried out by crashing the apps or any other mechanism to exhaust the resources. If the app is not designed and coded properly, the malicious user can make the app a node for a DoS attack to other apps.

"Elevation of Privileges" is that last category. The privileges can be escalated if the device is "jailbroken" or due to inadequate authentication and authorization methods. Adequate tests should be carried out to assess if the escalation of privileges can be performed to give a user high level of access than desired. For instance, tests can be conducted to assess if the privileges of a normal user can be escalated to the admin level.

4.4. App Testing Techniques

Mobile apps are pieces of software with mobile-specific elements. The auditing or testing of mobile apps involves similar techniques that are typically used in auditing of other software applications.

The application code can be audited to assess adherence to coding guidelines, or to identify instances of insecure coding principles, for example, an. assessment for buffer overflow vulnerability. OWASP's "Secure Coding Practices" publication includes input validation, output encoding, authentication management, session management, error handling and logging and other elements for perming coding in a secure manner. There are tools available for performing the evaluation of the code for many programming languages. For Java, one of the most widely used programming languages, the tools include Lint[12], Checkstyle[13] and PMD[14]. To audit the behavior of the application after executing the code, white-box testing or black-box testing is used. In white-box testing, the information about the code design and methods is known. This kind of testing is usually done by using the tools such as JUnit[15] or XCTest[16]. The white-box testing may include data flow testing and control flow testing. Black-box testing, on the other hand, is performed without any information about the code. Black-box testing is usually done to audit the behavior of the application without considering how the code has been written. The test cases are developed to audit the normal as well as particular conditions. The test cases can include boundary value tests, error guessing and cause-effect scenarios.

Although subjective in nature, the assessment report can also include an opinion on the usability of the application. There are certain accepted principles to enhance the perception about the usability of the application. The audit team can assess whether redundant or unnecessary text, buttons or other elements exist in the app. The app should

[12] http://tools.android.com/recent/lint

[13] http://checkstyle.sourceforge.net/

[14] http://pmd.sourceforge.net/

[15] http://junit.org/

[16] http://testng.org/doc/index.html

have a self-guiding interface. If the audit team assesses that a user would constantly be looking for more information to perform further actions, the observation can be included in the audit report. In general, the usability principles available at the resources such as Google Best Practices[17], , and "Usability.gov"[18] can be used for reference.

App auditing should also include the design and use assessment from the perspective of users with disabilities. For visually impaired individuals, the app should provide features such as a screen reader, font enlargement, the option to change the color scheme, a screen magnifier and the option to use voice recognition to utilize different features of the app. To assist people with auditory impairment, the app should have the ability to provide vibration or visual notification. If the app utilizes some video content, subtitles should be provided. Similarly, the app should have sufficient capabilities to assist individuals with physical or cognitive impairment. There are general accessibility guideless available for both Android[19] and iOS[20] platforms.

The assessment of the battery usage of the app includes performing testing on how the app is draining the battery. The battery performance is assessed by observing battery drainage behavior with and without installing the app. This is an important aspect of the evaluation because if the app drains significant battery resources, the users might not keep the app for long. Tools such as Fiddler[21] can be used to assess the requests made by the app to the device's backend system. The battery consumption testing should include both at battery's fully charged level as well as at the low charged level.

SQLite[22] database is used as the local database for most apps. The data stored in this local database is used when the device is not connected to the Internet. The testing of

[17] http://www.google.com/think/multiscreen/#mobile-best-practices

[18] https://www.usability.gov/what-and-why/index.html

[19] http://developer.android.com/tools/testing/testing_accessibility.html

[20] https://developer.apple.com/library/ios/technotes/TestingAccessibilityOfiOSApps/TestingtheAccessibilityofiOSApps/TestingtheAccessibilityofiOSApps.html

[21] http://www.telerik.com/fiddler

[22] http://www.sqlite.org/

the local database can be done using manual or automated techniques. All risks associated with a common database should also be audited for the app's local database. The tests include database stored procedures and function testing, validation testing, performance testing, integration testing, and security testing such as SQL injection testing, etc.

OWASP (Open Web Application Security Project) offers specific information for mobile security. The top ten mobile risks are[23]: weak server side controls, insecure data storage, insufficient transport layer protection, unintended data leakage, poor authorization and authentication, broken cryptography, client side injection, security decisions via untrusted inputs, improper session handling and lack of binary testing. While performing the testing, it should be ensured that the top risk areas identified by OWASP are adequately assessed.

5. Non-Technical Considerations – Intellectual Property

The audit scope for an app should also consider non-technical areas, including the review of basic tenets of intellectual property protection. This section includes points to consider for trade-secret, copyright and trademark protection. Furthermore, the app under assessment might use material from other sources. This section also provides an overview of the audit steps for assessment of appropriate permissions to reuse the material.

Any confidential information is a "trade secret" if it is not known in the industry, and may provide some economic benefits. A unique app developed for investors or beta testers can be classified under this category (Stim, 2010). For any information to be classified in this category, three criteria should be met. First, the information should not be "readily ascertainable." If others can obtain the information through publicly available domains, for example, through the internet or a published book, then the information cannot be considered a trade secret. Second, there should be reasonable evidence that the information under trade secret protection provides a competitive advantage or some

[23] https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

economic benefit. Lastly, the owner of the protected information has implemented reasonable procedures to prevent the disclosure of the information. Determining what procedures are reasonable to maintain secrecy is highly subjective. At the minimum, it includes restricting access through physical security principles, implementing appropriate logical security controls, requiring non-disclosure agreements (NDA) from anyone who has access to the information including the employees and third parties, and adequate information classification e.g. clearly label the information as "confidential." The audit should first consider whether the app, the code, or other related documents can be classified under trade secret or not. If it can, then the audit report should also contain an opinion on the three criteria mentioned above. The concept of trade secret applies only if the information, in fact, is secret. Information security professionals directly contribute to creating and maintaining the measures for trade secret protection. The audit team should assess that the appropriate information security team has engaged in such efforts. Furthermore, an extensive assessment should be performed to assess that NDA is available from anyone with the knowledge of the app, either the internal code, the technical design or the functionalities of the app (Stim, 2010). The NDA should, at minimum, contain some explanation of the trade secret that needs to be protected, and should explicitly designate responsibilities to keep the information secret and should indicate the length of applicable time for the agreement. The NDA should also include any applicable exclusion(s), for example, if the company voluntarily discloses the classified information then the receiving part is not obliged to respect the rights mentioned under the agreement. The auditors should be aware that defining reasonable security, as required by the third criteria mentioned above, is subjective. Balancing business requirements and the information security protection requirements can be a challenging task, and it should be the responsibility of management to make a final decision if the two requirements seem incompatible.

"Copyright" protection provides the legal owner of the app the right to prevent any third party from copying, distributing, or making amendments in the app. If the copyright has been registered for the app, the unauthorized use of the app can be stopped through court enforcement. In instances of dispute cases, three points need to be proved. First, there is a copyright registration of the app. Second, there is sufficient evidence that

someone copied the app, and lastly that the app is substantially similar to the one protected through the copyright registration. In the USA, the Digital Millennium Copyright Act (DMCA)[24] provides legislation regarding copyright protection for technological works. The audit scope should include assessment against the two key elements of the act. First, the app developers should not attempt to tamper with the copyrighted work. According to the section 1201 of DMCA, the Act imposes liability on those who circumvent technological measures that effectively control access to the protected work. DMCA defines circumvention as: "to descramble a scrambled work, to decrypt an encrypted work, or otherwise to avoid, bypass, remove, deactivate, or impair a technological measure, without the authority of the copyright owner." Second, the Act allows lawful reverse-engineering in cases if the computer program is legally acquired, and the reverse engineering is performed to analyze "those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs." [34]

"Trademark" is a name, design or slogan that is used to uniquely identify the source of goods. The terminology "Service Mark" is used to identify the source of services rather than the goods. The trademark or the service mark protects the right of a company to use the distinctive mark in the commerce activities. The purpose of the trademark legislation is to prevent "confusion" regarding the source of the goods or services. The trademark protection depends upon the "strength" classification system. Therefore, distinctive trademarks are relatively easy to protect whereas it might be difficult to prove that the consumers associate any generic description with the app. To demonstrate the association of generic description with the particular service, information about extensive marketing efforts is presented. To ensure that risks related to trademark protection are appropriately managed, the audit team should perform an assessment that demonstrates whether the app name is reasonably distinguished from others. Furthermore, if the app name is very similar to any other app providing the similar services, the audit report should include the assessment of trademark infringement risks. The audit team can simply search the app store or other software categories at the leading

[24] https://www.copyright.gov/legislation/dmca.pdf

vendors to assess if the app name is likely to be confused with any other product or services. Another aspect that needs to be considered for trademark assessment is "dilution." The dilution refers to the situation when there is likelihood that the distinction between another famous trademark will be blurred with a lesser-known trademark. For example, the trademark "Microsoft" will be considered diluted if someone creates an app titled "Macrosoft."

The auditors should perform an assessment that the appropriate permissions have been obtained before using someone else's work in the app. If only the small section of the work of others is included, it can be considered "fair-use." However, that determination is subjective, and in dispute cases, the judge might not agree that the use can be considered fair. Furthermore, no permission is necessary if the app contains information from public domain. The exemption from permission may include any material that was published before the intellectual property legislation came into effect. The auditors should be aware that any information posted on the internet cannot necessarily be considered that the information is in public domain. In cases where the permissions have been obtained, the auditors should assess that the permission identifies the needed rights (exclusive or non-exclusive), and the formal written agreements have been developed.

6. Conclusion

The technological advancements have significantly enhanced the capabilities of the mobile devices as the devices are now equipped with high processing powers and storage facilities. This, in turn, has created a huge market and appetite for mobile applications delivering a variety of functions. The popularity of mobile applications makes them an easy target for cyber criminals. Just like many other software developments, the mobile applications are sometimes developed with only functionality in mind. In certain cases, there is also pressure to release the app to the market as early as possible. The information security issues are not appropriately addressed, and the mobile applications become the threat vector for cybercriminals. To mitigate the risks associated with developing and deploying mobile applications, information auditing and security

professional should use a comprehensive methodology to assess the security posture of the app(s) used by their organizations. The methodology should incorporate both technical and non-technical elements so that adequate assurance about the risks associated with the app could be obtained. This paper presented background information on the points that need to be considered for an effective app auditing exercise. Since each app operates in a unique environment with a different combination of hardware and software components, it is essential that information presented in this paper be tailored towards the testing objectives of the app under testing.

Muzamil Riffat, muzamil@hotmail.com

7. References

Accenture (2013). *Electronics Products and Services Usage survey*, https://www.accenture.com/ae-en/~/media/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Technology_6/Accenture-Consumer-Electronics-Products-and-Services-Usage-Report.pdf

Apple (2017), https://developer.apple.com/app-store/freemium-business-model, Accessed April 2017

AppSec Labs. (2014). *Mobile application penetration testing*. Retrieved from https://appsec-labs.com/mobile_pentesting/

Armin, J. (2013). *Mobile threats and the underground Marketplace.* Retrieved from The Anti-Phishing Working Group website: http://docs.apwg.org/reports/mobile/APWG_Mobile_Report_v1.9.pdf

Boivie, Inger, Aborg, Carl, Persson, Jenny, Lofberg, Mats (2003): *Why usability gets lost or usability in in-house software development*. In: Interacting with Computers, Vol. 15, no 4

Compuware (2012), http://offers2.compuware.com/rs/compuware/images/Mobile_App_Survey_Report.pdf, Accessed April 2017

D'Orazio C., Choo K.K.R. (2015) *A generic process to identify vulnerabilities and design weaknesses in iOS healthcare apps.* In: System Sciences (HICSS), 48th Hawaii International Conference on; IEEE; 2015:5175–5184.

D'Orazio C., Choo K.K.R. An adversary model to evaluate DRM protection of video contents on iOS devices. J. Comput. Secur. 2016; 56:94–110.

Mu:

Enderle, R. (2012a). *McAfee shows security flaws of smartphones (especially android devices)*. CIO. Retrieved from http://www.cio.com/article/720001/McAfee_Shows_Security_Flaws_of_Smartphones_Especially_Android_Devices_?page=2&taxonomyId=3067

Farnden J., Martini B., Choo K.K.R. *Privacy risks in mobile dating apps*. In: Proceedings of 21st Americas Conference on Information Systems, AMCIS; Association for Information Systems; 2015. http://aisel.aisnet.org/amcis2015/ISSecurity/GeneralPresentations/13.

Faruki P., Bharmal A., Laxmi V., Ganmoor V., Gaur M.S., Conti M., Rajarajan M. *Android security: a survey of issues, malware penetration, and defenses*. IEEE Commun. Surv. Tutorials. 2015;17(2):998–1022.

FbombMedia (2017), https://fbombmedia.com/freemium-paid-ad-supported-app-monetization-strategy-right/, Accessed April 2017

Gartner (2013), http://www.gartner.com/newsroom/id/2592315, Accessed by April 2017

Hayes, R. & Rangarajan, K. (2012) *10 tips for mobile application security* [brochure]. Retrieved from Dell SecureWorks website: http://www.secureworks.com/assets/pdfstore/brochures/Dell_10_Tips_for_Mobile_App_Security_for_Retailers.pdf

Knott, Daniel (2015), Hands-On Mobile App Testing: A Guide for Mobile Testers and Anyone Involved in the Mobile App Business, Addison-Wesley Professional (ISBN: 9780134191829)

Lange, M., Liebergeld, S., Liebergeld, A., Warg, A., & Peter, M. (2011). L4Android: A generic operating system framework for secure smartphone. pp.1-12 Retrieved from

https://www.isti.tu-berlin.de/fileadmin/fg214/liebergeld/spsm03-lange.pdf

Mobilefuture (2017), http://mobilefuture.org/issues/internet-of-things/, Accessed April 2017

Mobile Iron (2015), https://www.mobileiron.com/en/quarterly-security-reports/q4-2015-mobile-security-and-risk-review, Accessed April 2017

Nielsen (2014), http://www.nielsen.com/us/en/reports/2014/an-era-of-growth-the-cross-platform-report.html, Accessed April 2017

OpenSignal (2015), https://opensignal.com/reports/2015/08/android-fragmentation/, Accessed April 2017

OWASP, https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf

Pew Research Center Technology Ownership Survey (2015), http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015/, Accessed April 2017

Siciliano, R. (2012, June 13). *How does jailbreaking or rooting affect my mobile device security* [Web log post]. Retrieved from McAfee website http://blogs.mcafee.com/consumer/howdoes-jailbreaking-or-rooting-affect-my-mobile-device-security

Stim, Richard (2010), *Protecting Your Mobile App IP: The Mini Missing Manual*, Publisher: O'Reilly Media, Inc, (ISBN: 9781449393670)

Walberg, R. (2013, March 14). *Proactively and reactively protecting proprietary data*. Financial Post. Retrieved from http://business.financialpost.com/2013/03/14/proactively-andreactively-protecting-proprietary-data

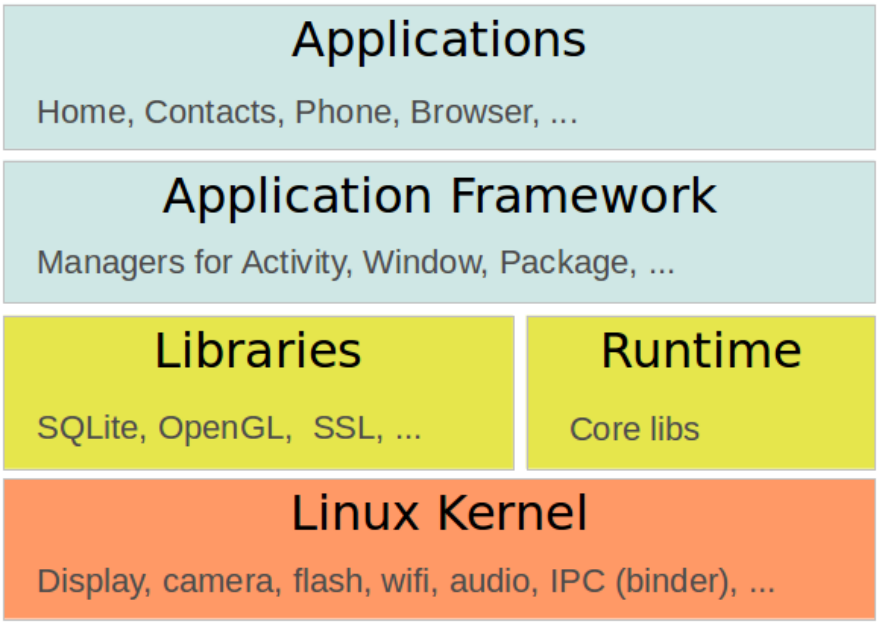
Zielinski, D. (2012, February). Bring your own device: More employers are allowing employees to use their own technology in the workplace. HR Magazine, 57(2), 71-74

Muzamil Riffat, muzamil@hotmail.com

8. Appendices

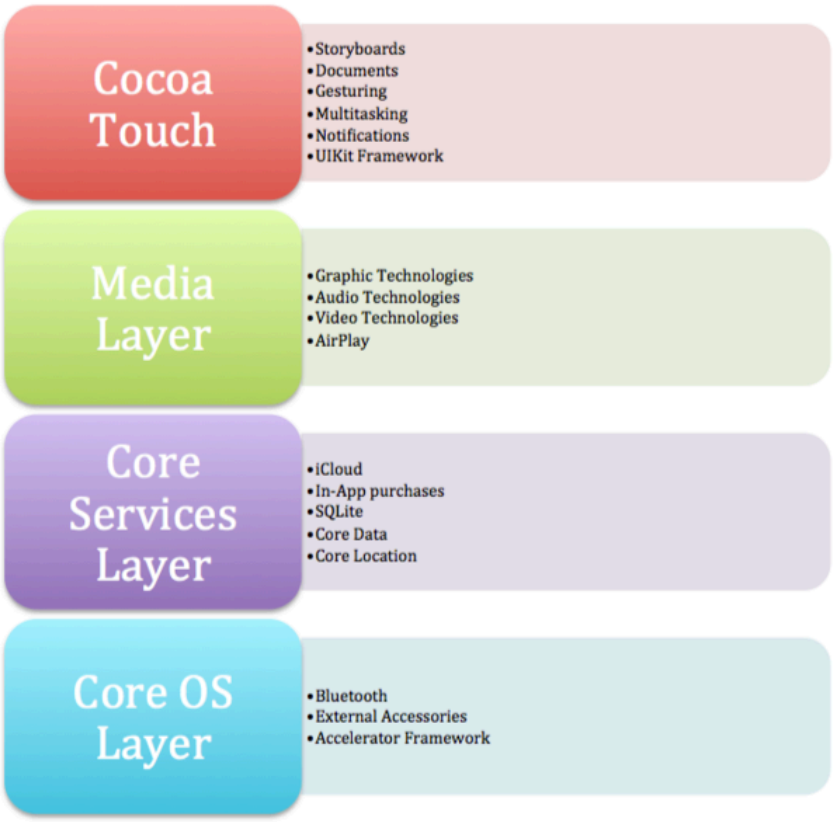
Appendix A:

The following picture shows different layers of Android architecture.



Source: http://www.vogella.com/tutorials/Android/article.html

The following picture shows different layers of iOS architecture.



Source: https://tilakgondi.wordpress.com/2015/01/14/ios-architecture/

Muzamil Riffat, muzamil@hotmail.com