Mobile Application Security

Best Practices to Optimize Security

July 2014

White Paper

Email: bdg@qburst.com | Website: www.qburst.com

Contents

Overview

Security leaks and confidential data disclosure from web and mobile apps are quite common today. With the increasing number of technologically rich mobile applications hitting the market, mobile phones have become the new target for hackers.

Android is one of the most popular mobile phone operating systems and is claimed to hold more than 36% of the market share [1]. Due to its popularity, Android is more prone to attacks.

Objective

This white paper elucidates the necessity of security testing mobile applications, the major threats that mobile applications are susceptible to, methodologies and tools used for mobile application security testing, best practices to create a robust mobile app, and some important guidelines for users and developers.

Audience

- Testers who want to specialize in mobile application security testing
- Developers can refer to this white paper to develop secure applications
- Mobile phone users can understand threats and learn how to protect themselves from attacks

Scope

The paper covers security testing of Android applications. It does not include mobile application development, application installation or similar areas.

Definitions

- **OWASP:** Open Source Web Application Security Project
- **Qasat:** Tool to help static analysis of Android apps
- **HashQ:** Tool to help find manipulated Android apps
- **WebScarab:** An intercepting proxy used to observe communication between two sides
- **WebSlayer:** A fuzzing tool used to brute force
- **IMEI:** The International Mobile Station Equipment Identity is a number, usually unique, used to identify 3GPP (GSM, UMTS and LTE) and iDEN mobile phones, as well as some satellite phones.
- **IMSI:** The International Mobile Subscriber Identity is used to identify the user of a cellular network
- **UDID:** Unique Device Identifier
- **MITRE:** Not for profit organization that operates FFRDCs (Federally Funded Research and Development Centers)
- **PCI DSS:** Payment Card Industry Data Security Standard
- **DISA:** Data Interchange Standards Association (http://www.disa.org)
- **FTC:** Federal Trade Commission (http://www.ftc.gov)

What is Mobile Application Security Testing?

Mobile phone usage is growing by the day. Unlike the situation a decade ago, today, people feel handicapped and uncomfortable without their mobile device close at hand. There have been great advances in mobile computing. People can download apps that help them socialize, keep fit, get directions, transact, shop, and much more. There are millions of mobile applications available in app stores that make our simple life simpler.

Amidst all the great things that have been accomplished in the mobility space, there is a global community of hackers who have been watching the mobile space closely. They use newer and bolder techniques to break into mobiles and applications, so app developers need to be cautious.

Mobile applications security testing is the process of reviewing the application characteristics and the code for vulnerabilities. It is a combination of static analysis, code review, and penetration testing.

Applications Become Prone to Attacks When:

1. Security flaws originate at the development stage
2. An application is cloned with extra functions that run in the background and perform malicious actions

In both cases, the application can compromise sensitive information contained in user devices. Security testing involves analysis to find out if the application is vulnerable. The application is dissected into component code and assets. The code is checked for flaws and penetration testing is done to determine whether unauthorized access is possible.

OWASP

The Open Web Application Security Project (OWASP) is a worldwide nonprofit charitable organization focused on improving the security of software. OWASP is involved in detecting and combating leaks in application security and techniques. They provide testers and developers guidelines to create secure applications.

OWASP Top Ten

The OWASP Top Ten is a list of vulnerabilities determined by identifying some of the most critical risks faced by mobile platforms. The OWASP Top 10 is referenced by many standards, books, tools, and organizations such as MITRE, PCI DSS, DISA, FTC, and others.

The OWASP Top Ten Highlights the following Threats to Mobile Applications:

1. Insecure Data Storage
2. Weak Server-Side Controls
3. Insufficient Transport Layer Protection
4. Client Side Injection
5. Poor Authorization and Authentication
6. Improper Session Handling
7. Security Decisions via Untrusted Inputs
8. Side Channel Data Leakage
9. Broken Cryptography
10. Sensitive Information Disclosure

In security analysis, priority is given to the user and user data, which is targeted the most. Vulnerabilities that pose a threat to user data are a security concern.

Insecure Data Storage

Insecure data storage, as the name suggests, means improper storage of data. Mobile applications should not store any data on the device unnecessarily.

According to OWASP [2]

"*Insecure data storage, occurs when development teams assume that users will not have access to the phones file system and store sensitive pieces of information in data-stores on the phone. Devices file systems are often easily accessible and you should expect a malicious user to be inspecting your data stores. Rooting or jailbreaking a device usually circumvents any encryption protections and in some cases, where data is not protected properly, all that is needed to view application data is to hook the phone up to a computer and use some specialized tools.* "

Data stored in devices can be easily extracted, and therefore needs to be properly secured. Recent reports show that even encrypted phones can be exploited. [3]

Weak Server-Side Controls

The server-side of any mobile application is a web application. When the server-side validation is not done properly, the server is prone to attacks. This in turn could put the client application and device at risk.

In a client-server hierarchy, where the mobile device is the client end, attacks intended to damage the server can be initiated from the device. If the server-side is not validated properly, it could result in a security breach.

There are other ways to attack the server-side. In case of Android applications, an emulator can be used. Browser-based attacks can also be generated from big browsers using a modified user agent.

Insufficient Transport Layer Protection

The most important feature of client-server architecture is information exchange. When data is exchanged, information may be exchanged through the carrier network or the Internet. While developing an application, if care is not taken while sharing data between the client and server, there is a chance that the data may be compromised in transit. The best way to protect data in transit is to encrypt it. This prevents data from being sniffed, particularly in the case of usernames, passwords, and credit card information.

According to OWASP
"*Unfortunately, mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication, but not elsewhere, exposing data and session IDs to interception. In addition, the existence of transport security does not mean it is implemented to its full potential. Detecting basic flaws is easy. Just observe the phone's network traffic. More subtle flaws require inspecting the design of the application and the applications configuration.*"

Even though the exploitability level is difficult, any threat to data should be taken seriously.

Client-Side Injection

Client-side injection refers to application vulnerabilities in the absence of client-side validation. The local storage of mobile devices is targeted here. Attacks such as SQL injections become very dangerous if the application has multiple user accounts on the same device. For example, a shared device may contain more than one user account in the database, which, if compromised, can lead to exposure of sensitive data.

All client-side injections that affect a web application can also pose a threat to the mobile application. The ease of exploitability of a client-side injection in vulnerable applications is the biggest risk. In case of mobile applications, anyone with physical access to the phone can generate an attack.

Prevention of client-side injection can be ensured by using proper validation methods.

Poor Authorization and Authentication

Authentication and authorization refers to user privileges granted for using an application. In an application with functionalities beyond publicly usable features, permission may be required for accessing privileged functions. Authentication refers to who you are in an application, and

authorization points to what you are authorized to do in an application. When the authorization and authentication schema fails to protect the application, the privileged functions in the application are compromised, rendering it vulnerable to attacks.

Authorization and authentication should be dealt with meticulously while developing an application to ensure that unauthorized users are not granted access to sensitive information. This can be achieved by ensuring secure session-handling and login functions.

Improper Session Handling

The sessions used in mobile devices are much longer compared to web applications. The session identifier and the session mechanisms used in mobile devices can threaten valid sessions of users. Sometimes the application chooses the device ID as the session identified because of its authenticity. However, the application ID can be extracted using third-party software, in which case the session ID can be made available to third parties as well, exposing valid sessions to attacks.

Encryption of valid session data is highly recommended. The application should use encryption in order to protect valid session data from being stolen through LAN sniffing and Wi-Fi attacks.

Security Decisions via Untrusted Inputs

Security decisions made in an application should not solely depend on user inputs. If security-related decisions are based on user inputs, it could leave the application vulnerable to threats. In case of mobile devices, security decisions should be taken with utmost care. Testing should be done on both sides of the application – server-side as well as client-side.

The application's logic should ensure that security does not depend on the user input, which could be abused to bypass permissions and security models. In a platform such as Android, the intent may be abused to attack applications. [4]

Side Channel Data Leakage

Side channel data leakage is considered to be a wide area of attack. The data contained in the system logs, and data that can be derived from leaks, power consumption, and timing information comprise side channel data leakage. These provide additional information that can be exploited if not protected.

In a Side Channel Data Leakage Attack, Data Is Obtained from the following Sources: [5]

- Web caches
- Keystroke logging
- Screenshots
- Logs
- Temp directories

Third-party libraries and APIs used in application development may contain malicious code that could steal data stored by the application. Developers need to ensure that only trusted APIs and libraries are used in the application. While executing procedures such as logging, sensitive data should not appear in the logs.

Case studies have shown that some applications store a snapshot of the running state when it is interrupted. This snapshot can be used to make the user believe that the application is working well even when it is interrupted. Such practices can lead to snapshots being stored insecurely leading to compromised data.

Broken Cryptography

Cryptography is the art of protecting information by transforming it into an unreadable format called cipher text. Cipher text may not resemble the original text; however, the original text can be decrypted from the encrypted data/cipher text.

Cryptography is used in mobile applications to ensure that the application stores the user data securely. Encryption is also used when applications fail to provide proper security to the data.

This Failure Is Caused Due to Two Reasons:

1. Broken implementations of strong algorithms: The developer may use a strong algorithm to protect the data, but the implementation of the algorithm is flawed, resulting in the data being decrypted by a hacker.
2. Weak algorithm or crypto implementation: Instead of using a strong algorithm, the developer uses a weak algorithm that can be decrypted effortlessly.

In both cases, the application fails to protect sensitive user data. In many cases, developers confuse between encryption and other text scrambling methods. They fail to realize that encoding, obfuscation, and serialization is not encryption.

Sensitive Information Disclosure

Sensitive information disclosure refers to the personal and important data stored in the application as hardcoded values such as passwords and credit card information. Applications that are developed for mobile devices can be reverse engineered, revealing the code of the application. Sensitive information hardcoded in the application can be accessed by hackers using this technique. The sensitive information thus gained can be used to craft attacks.

Types of Mobile Application Security Testing

We Can Divide Mobile Application Testing into Three Parts:

1. Dynamic analysis
2. Black box security testing
3. Static analysis & code review

Dynamic Analysis

In dynamic analysis, the behavior of an application is analyzed after installing it on various versions of compatible and non-compatible devices. This method of testing helps testers understand possible flaws. The behavior of the application is observed and test cases are created based on observations.

Black Box Security Testing

Black box testing involves treating the application like a black box that produces output to input stimuli. The tester feeds the application with inputs and observes the response. The input strings for the application are crafted based on the results of dynamic analysis and the OWASP testing methodology. The application is tested thoroughly with several well-crafted attacks to make sure the application can defend itself.

Static Analysis & Code Review

Static analysis and code review covers the analysis of the application code and coding defects. The code of the application is analyzed using static analyzers. The code is reviewed manually and checked for vulnerabilities that may arise due to poor coding practices.

With static analysis, the business logic and the security of the application are covered. The code reviewer tests the application for each taint location in the application.

Identifying and Protecting Data in Mobile Devices

Identification and protection of sensitive data on a mobile device is an important aspect of mobile application development. Mobile devices are expected to replace desktop computers in the near future. With improvements in technology, mobile devices have managed to scale up to match the complex computing capabilities of a desktop computer. From management of email account credentials to control of automobiles, mobile phones can do it all today.

Hackers attack an application with the intention to derive certain value. The more valuable the target is, the more prone it is to attacks. For developers, the data contained in the application is valuable. Every piece of data that the application handles is significant and developers must handle it with care. Protection of data should be one of the primary goals of a mobile developer.

The close interaction with an operating system allows the application on devices to access more data than a browser on a computer. This helps to build innovative features into the application. The data is vulnerable when the entire application is not developed by the same team. There may be APIs and code from other sources; some applications tend to be interconnected. These possibilities cause the data in the application to leak. Additionally, third-party services such as advertising are gaining ground. If these services are not integrated carefully, it can lead to disclosure of significant amounts of personal data.

Mobile users are ignorant about the technical functionality of an application and assume that application data is secure. Users usually install applications without verifying source and remain clueless about how the application processes data. Users remain unaware even when the application makes an external connection to the internet to process the data. Hackers take advantage of user ignorance to exploit data on the devices.

How to Identify Sensitive Data?

Every piece of data is sensitive. Data cannot be classified as sensitive and non-sensitive. Users enter data into an application under the assumption that security will not be compromised. Considering the importance users give to data, applications should be designed to treat every little piece of user data as sensitive.

Examples of Personal Data Users Prefer to Keep Private:

- Their location
- Contacts
- Unique device and customer identifiers (such as IMEI, IMSI, UDID and phone number)
- Identity of the data subject
- Identity of the phone (make of the phone)
- Credit card and payment data
- Phone call logs, SMS or instant messaging
- Browsing history

- Email
- Information society service authentication credentials (especially services with social features)

Protecting Data—Things to Remember

- The data handled by an application should be protected from storage to transit
- Access to data being stored in another field is to be taken into consideration while handling data
- An important location where data leak can occur is the side channel data leakage
- Data should be logged or shown in error logs
- Each piece of code that handles data needs to be crafted carefully
- User data should be encrypted using smart algorithms before being stored on the device
- The encryption method should use a strong key
- The data stored on the device should be accessible only to the application that stores the data
- The data should not be given global read privileges leading to other applications residing on the device
- Whenever the data is transferred to other locations, such as a server, the application should use https

Creating a Formidable App

Developers creating mobile applications need to realize that the mobile application is only a part of the system that attackers target. When an application is built, every piece of information that enters the application needs to be validated. Some points that should be taken into consideration while developing a formidable app are:

- User input should be considered, but not enforced while making security decisions
- The data stored on the device should be handled carefully to ensure that none of the information is accessible even when the device changes hands
- The permissions set to the files and databases should ensure that application use is unique and should be accessible only to the owner
- The user may install a malicious application accidentally. Such applications should not be able to access the files and database of the developed application (Malicious applications enter the device when the user installs application from untrusted sources)

Steps to Create a Secure and Powerful Application:

The first step is to identify the data that is most critical to an application or a device – this can be done by threat modeling the data before development. Consider all the data that the application uses, analyze the data and identify the threat level associated with the data. Once the threat modeling is done, decide the level of security that is required to protect the data.

Next, implement the level of security required to protect the data. During the coding phase the developer writes necessary protection methods for the data. This includes validations on both client and server sides to hashing and encryption of data. Security is embedded into the application without disturbing the business logic of the application. The application works normally with all the security features embedded into it.

The third phase of the process is the testing of the application to verify all the implemented security and application logic. Once the application is complete, it is subjected to thorough testing to analyze the quality of the application.

Security Testing Tools

Qasat

Qasat is an Android static analyzer. The application helps code reviewers decompose an Android Package File (APK) and understand the application better. The analyzer decomposes an APK file into its components. The assets in the applications are enumerated as lists. Qasat also enumerates code fragments that are considered sensitive. Qasat allows the user to save the code into a location of their choice. This helps the code reviewer to review the code.

HashQ

HashQ is a tool used to list out difference between two applications of the same type (JAR/APK). Built on pure shell scripts and the GUI called Yad, HashQ highlights the file-level differences between the applications under analysis.

Android Emulator

Android Emulator is an application that emulates and tests virtual android devices. Applications can be installed on virtual machines and used as if they are installed on a real device. Emulator is free to download and easy to install. The emulator is also included in the Android SDK.

WebScarab

WebScarab is an intercepting proxy used to monitor conversations between a client and server. Intercepting proxy is used to watch the outgoing and incoming traffic from a device. The conversations and server headers from responses can be viewed in detail.

WebSlayer

WebSlayer is a tool used to fuzz applications to test different values for parameters. This helps a tester to test a large number of parameters at a time.

Other Penetration Testing Tools

There are lot of other penetration testing tools (Fuzzers, Crawlers, Spiders and others) and scripts that are used to attack applications.

Real-Time Examples

Wikipedia defines a user as "*an agent, either a human agent (**end-user**) or software agent, who uses a computer or network service.*" (http://en.wikipedia.org/wiki/User_(computing))

Most users download and use the service, and remain clueless about the technicalities involved in development. The only thing that most users care about while using an application is its utility. Any additional feature or functionality is a plus.

In reality, a mobile application can perform any function desired from it without the end user's knowledge. Malicious applications get installed on the device without warning. In many cases, the web application that provides the user with the installer is not legitimate.

There are stores from which users can download applications. Google Play is an example of a legitimate store. The applications in such stores are subjected to thorough code review and analysis and suspicious apps are barred.

In most cases, users are not aware of the damage malicious applications can cause. From sending premium SMSs to leaking mails and contacts details, a malicious application is capable of plenty of damage.

There are numerous examples of malicious clones of well-known applications available on third-party sites.

Rogue Instagram Application

The fake Instagram application was distributed through a Russian website. The application was designed to send SMS to premium numbers yielding revenue to its creators. The scam was so convincing that users were unable to distinguish between the fake and real app. Suspicions arose only when cell phone bills spiked.

The Trojan contained in the application was identified as Andr/Boxer-F.

Image courtesy: http://sophosnews.files.wordpress.com/2012/04/fake-instagram-web1.jpg?w=640

The permissions that the application uses are different from the real Instagram application.

Image courtesy: http://sophosnews.files.wordpress.com/2012/04/instagram-malware-permissions.jpg?w=640

DroidDream

DroidDream is a Trojan attack that is hidden within legitimate looking Android apps. It is considered more intrusive than other malicious components as it has broken into Android application stores.

DroidDream lives up to its name. The application is designed to infect the device during sleep time – between 11 PM and 8 AM.

Once the infected device is rooted, DroidDream searches for the package "com.android.providers.downloadsmanager." If the package is not installed, DroidDream installs a second malicious app. Other malicious applications can be installed from the DroidDream command and control servers. DroidDream passes on a lot of information to the command and control center. This includes IMEI, IMSI, device model, SDK version, language, country, and user ID.

[Image courtesy: https://blog.lookout.com/wp-content/uploads/2011/03/screenshot_super_guitar_solo_small.png]

Some of the other malicious applications are listed below. Some of them are based on the Trojan DroidDream.

- Falling Down
- Super Guitar Solo
- Super History Eraser
- Photo Editor
- Super Ringtone Maker
- Super Sex Positions
- Hot Sexy Videos
- Chess
- Livelocker
- TDT direct to TV
- Mera live TV
- PhotoShop tutorials Free
- Dealers Scale lite
- Cute wallpapers
- Beach wallpapers
- Android video converter
- TV by Zurera
- owling Time
- Advanced Barcode Scanner
- Super Bluetooth Transfer
- Task Killer Pro
- Music Box
- Sexy Girls: Japanese
- Sexy Legs
- Advanced File Manager
- Magic Strobe Light

Conclusion

Android's relevance today cannot be ignored. The popularity of the platform leaves it susceptible to attack. Android applications need to be developed taking into consideration different kinds of security concerns.

The application should follow proper threat modeling and secure application coding guidelines recommended by organizations such as OWASP.

Key Takeaways:

- The data handled by the application should be managed securely
- The transfer of data and logging of information should be done in a secure and controlled manner
- Application architecture should be subject to threat modeling before development
- The sensitive areas of the application should be identified and proper methods should be adopted to ensure that the data is protected
- Every application should be security tested before release
- App users should be cautious while downloading applications from third party sites

Applications leak data in many ways. In many cases application developers fail to properly handle threats. A well-built application coupled with user discretion can guarantee great user experience.

References

[1] Wikipedia: http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

[2] Source: https://www.owasp.org/index.php/Mobile_Top_10_2012-M1_Insecure_Data_Storage

[3] http://www.extremetech.com/computing/150536-how-to-bypass-an-android-smartphones-encryption-and-security-put-it-in-the-freezer

[4] Source: http://image.slidesharecdn.com/appsecusa2011-owasptop10mobilerisks-1-110927114100-phpapp02/95/slide-29-728.jpg?1317141936

[5] Source: http://image.slidesharecdn.com/appsecusa2011-owasptop10mobilerisks-1-110927114100-phpapp02/95/slide-32-728.jpg?1317141936

Email: bdg@qburst.com | Website: www.qburst.com