CHAPTER

1

Mobile Application (In)security

There is little doubt that mobile computing has changed the world; in particular, the way you work, interact, and socialize will never be the same again. It has brought infinite possibilities to your fingertips, available all the time. The ability to do your online banking, check your e-mail, play the stock market and much, much more are just a swipe away. Indeed, application development is now so popular that Apple's trademark, "There's an app for that" is bordering on reality.

This chapter takes a look how mobile applications have evolved and the benefits that they provide. It presents some metrics about the fundamental vulnerabilities that affect mobile applications, drawn directly from our experience, demonstrating that the vast majority of mobile applications are far from secure. We then examine a means to categorize these vulnerabilities based on the Open Web Application Security Project (OWASP) Top 10 mobile security risks. We also provide a high-level overview of some of the open source mobile security tools endorsed by OWASP, how you can use them to identify some of the issues detailed in the project, and where to find them. Finally, we describe the latest trends in mobile application security and how we expect this area to develop in the future.

The Evolution of Mobile Applications

The first mobile phone applications were developed by handset manufacturers; documentation was sparse, and little information existed in the public domain on the operating internals. This can perhaps be attributed to a fear from the vendors that opening the platforms to third-party development might have exposed trade secrets in what was not yet a fully developed technology. The early applications were similar to many of the manufacturer-based apps found on today's phone, such as contacts and calendars, and simple games such as Nokia's popular *Snake*.

When smartphones emerged as the successor to personal digital assistants (PDAs), application development really began to take off. The growth of mobile applications can perhaps be directly attributed to the increased processing power and capabilities of the smartphone combined with the growing demand for functionality driven by the consumer market. As smartphones have evolved, mobile applications have been able to take advantage of the enhancements of the platforms. Improvements in the global positioning system (GPS), camera, battery life, displays, and processor have all contributed to the feature-rich applications that we know today.

Third-party application development came to fruition in 2008 when Apple announced the first third-party application distribution service, the App Store. This followed on from the company's first smartphone, the iPhone, which had been released the previous year. Google closely followed with the Android Market, otherwise known today as Google Play. Today, a number of additional distribution markets exist, including the Windows Phone Store, the Amazon Appstore, and the BlackBerry World to name but a few.

The increased competition for third-party application development has left the developer markets somewhat fragmented. The majority of mobile applications are platform specific, and software vendors are forced to work with different operating systems, programming languages, and tools to provide multi-platform coverage. That is, iOS applications traditionally have been developed using Objective-C, Android, and BlackBerry applications using Java (up until BlackBerry 10, which also uses Qt) and Windows Phone applications using the .NET Framework. This fragmentation can often leave organizations requiring multiple development teams and maintaining multiple codebases.

However, a recent increase has occurred in the development of cross-platform mobile applications as organizations look to reduce development costs and overheads. Cross-platform frameworks and development of HTML5 browser-based applications have grown in popularity for these exact reasons and, in our opinion, will continue to be increasingly adopted.

Common Mobile Application Functions

Mobile applications have been created for practically every purpose imaginable. In the combined Apple and Google distribution stores alone, there are believed to be more than 2 million applications covering a wide range of functions, including some of the following:

- Online banking (Barclays)
- Shopping (Amazon)
- Social networking (Facebook)
- Streaming (Sky Go)
- Gambling (Betfair)
- Instant Messaging (WhatsApp)
- Voice chat (Skype)
- E-mail (Gmail)
- File sharing (Dropbox)
- Games (*Angry Birds*)

Mobile applications often overlap with the functionality provided by web applications, in many cases using the same core server-side APIs and displaying a smartphone-compatible interface at the presentation layer.

In addition to the applications that are available in the various distribution markets, mobile applications have been widely adopted in the business world to support key business functions. Many of these applications provide access to highly sensitive corporate data, including some of the following, which have been encountered by the authors during consultancy engagements:

- Document storage applications allowing users to access sensitive business documents on demand
- Travel and expenses applications allowing users to create, store, and upload expenses to internal systems
- HR applications allowing users to access the payroll, time slips, holiday information, and other sensitive functionality
- Internal service applications such as mobile applications that have been optimized to provide an internal resource such as the corporate intranet
- Internal instant messaging applications allowing users to chat in real time with other users regardless of location

In all of these examples, the applications are considered to be "internal" applications and are typically developed in-house or specifically for an organization.

Therefore, many of these applications require virtual private network (VPN) or internal network access to function so that they interact with core internal infrastructure. A growing trend in enterprise applications is the introduction of "geo fencing" whereby an application uses the device's GPS to ascertain whether a user is in a certain location, for example, the organization's office, and then tailors or restricts functionality based on the result.

Benefits of Mobile Applications

It is not difficult to see why mobile applications have seen such an explosive rise in prominence in such a short space of time. The commercial incentives and benefits of mobile applications are obvious. They offer organizations the opportunity to reach out to end users almost all the time and to much wider audiences due to the popularity of smartphones. However, several technical factors have also contributed to their success:

- The foundations of mobile applications are built on existing and popular protocols. In particular, the use of HTTP is widely adopted in mobile deployments and is well understood by developers.
- The technical advancements of smartphones have allowed mobile applications to offer more advanced features and a better user experience. Improvements in screen resolution and touch screen displays have been a major factor in improving the interactive user experience, particularly in gaming applications. Enhancements in battery life and processing power allow the modern smartphone to run not just one but many applications at once and for longer. This is of great convenience to end users as they have a single device that can perform many functions.
- Improvements in cellular network technologies have resulted in significant speed increases. In particular, widespread 3G and 4G coverage has allowed users to have high-speed Internet access from their smartphones. Mobile applications have taken full advantage of this to provide access to an array of online services.
- The simplicity of the core technologies and languages used in mobile development has helped with the mobile revolution. Applications can be developed using popular and mature languages such as Java, which are well understood and have a large user base.

Mobile Application Security

Mobile applications are affected by a range of security vulnerabilities, many of which are inherited from traditional attacks against web and desktop applications. However, several other classes of attack are specific to the mobile area and

arise due to the way in which mobile applications are used and the relatively unique entry points and the attack surfaces that these apps create. Consider the possible attack surfaces for a mobile application that developers should be aware of and look to defend against:

- Most mobile applications perform some kind of network communication, and due to the nature in which mobile devices are used, this communication may often occur over an untrusted or insecure network such as hotel or café Wi-Fi, mobile hotspot, or cellular. Unless data is adequately secured in transit, it may expose an application to a number of possible risks, including disclosure of sensitive data and injection attacks.
- Mobile devices are carried with you wherever you go, creating many opportunities for them to be lost or stolen. Mobile application developers must recognize the risks from data recovery attempts against a device's filesystem. Any residual content that an application leaves on the filesystem, whether it's through persistent storage or temporary caching, can potentially expose sensitive data to an attacker.
- A scenario that is fairly unique to mobile applications is awareness of threats originating from the host device. Malware is rife within the mobile space, particularly in the unofficial distribution markets, and developers must be conscious of attacks from other applications.
- Mobile applications can derive input from a large number of possible sources, which creates a significant number of possible entry points. For example, seeing applications accept data from one or many of the following is not uncommon: near field communication (NFC), Bluetooth, camera, microphone, short message service (SMS), and universal serial bus (USB) or quick response (QR) codes to name but a few.

The most serious attacks against mobile applications are those that expose sensitive data or facilitate a compromise of the host device. These vulnerabilities are more often than not limited to the mobile end user's data and device as opposed to all users of the service. Although server-side vulnerabilities pose the greatest risk to mobile application deployments as a whole because they can expose unrestricted access to back end systems, these issues are well documented and understood. Server-side vulnerabilities in mobile applications are not covered in the context of this book; however, we highly recommend *The Web Application Hacker's Handbook* (`http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118026470.html`) if you would like to know more about this attack category.

Mobile application security is still somewhat misunderstood and has not fully matured as an area of focus; indeed, the majority of mobile applications are still considered insecure. We have tested hundreds of mobile applications in recent years and one or more serious security issues affected the majority of them. Figure 1-1 shows what percentage of these mobile applications tested

since 2012 were found to be affected by some common categories of client-side vulnerability:

- **Insecure data storage (63%)**—This category of vulnerability incorporates the various defects that lead to an application's storing data on the mobile device in either cleartext, an obfuscated format, using a hard-coded key, or any other means that can be trivially reversed by an attacker.
- **Insecure transmission of data (57%)**—This involves any instance whereby an application does not use transport layer encryption to protect data in transit. It also includes cases where transport layer encryption is used but has been implemented in an insecure manner.
- **Lack of binary protections (92%)**—This flaw means that an application does not employ any form of protection mechanism to complicate reverse engineering, malicious tampering, or debugging.
- **Client-side injection (40%)**—This category of vulnerability describes scenarios where untrusted data is sent to an application and handled in an unsafe manner. Typical origins of injection include other applications on the device and input populated into the application from the server.
- **Hard-coded passwords/keys (23%)**—This flaw arises when a developer embeds a sensitive piece of information such as a password or an encryption key into the application.
- **Leakage of sensitive data (69%)**—This involves cases where an application unintentionally leaks sensitive data through a side channel. This specifically includes data leakages that arise through use of a framework or OS and occur without the developer's knowledge.

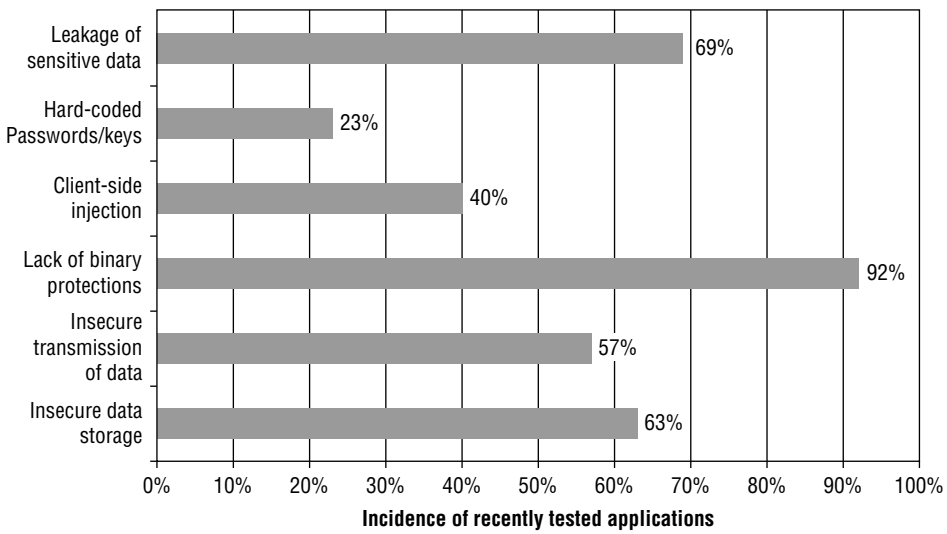

Figure 1.1: The incidence of some common mobile application vulnerabilities recently tested by the authors

Key Problem Factors

The core security problems in mobile applications arise due to a number of factors; however, vulnerabilities typically occur when an application must handle or protect sensitive data or process data that has originated from an untrusted source. However, several other factors have combined to intensify the problem.

Underdeveloped Security Awareness

Unlike most web applications where the attack surface is limited to user-derived input, mobile application developers have a number of different scenarios to consider and protect against. Mobile application development is fairly unique when compared to the development of other applications in that developers cannot trust the host operating system or even their own application. Awareness of the many attack surfaces and defensive protections is limited and not well understood within the mobile development communities. Widespread confusion and misconceptions still exist about many of the core concepts involved in mobile security. A prime example is that many developers believe that they don't need to encrypt or protect data that is persistently stored on the device because it is encrypted through the data-at-rest encryption that comes standard with many devices. As you will discover, this assumption is not accurate and can expose sensitive user content.

Ever-Changing Attack Surfaces

Research into mobile device and application security is a continually evolving area in which ideas are regularly challenged and new threats and concepts discovered. Particularly on the device side, discovering new vulnerabilities that may undermine the accepted defenses that an application employs is common. A prime example of this was the discovery of Apple's "goto fail" vulnerability (`http://support.apple.com/kb/HT6147`), which undermined the integrity of what was previously believed to be a secure communications channel. In this instance even recommended protections such as certificate pinning could be bypassed, which lead to many developers and security professionals researching and implementing secondary encryption schemes to protect data inside the SSL/TLS channel. These types of vulnerabilities demonstrate how on-going research can affect or change the threat profile for an application even partway through a development project. A development team that begins a project with a comprehensive understanding of the current threats may have lost this status and have to adapt accordingly before the application is completed and deployed.

Economic and Time Constraints

Most application development projects are governed by strict resource and time constraints, and mobile application development is no exception. The economics of an application development project often mean that having permanent security

expertise throughout the development process is infeasible for companies, particularly in smaller organizations that on the whole tend to leave security testing until late in a project's lifecycle. Indeed, smaller organizations typically have much smaller budgets, which means they are often less willing to pay for expensive security consulting. A short time-constrained penetration test is likely to find the low-hanging fruit, but it is likely to miss more subtle and complex issues that require time and patience to identify. Even in projects with a permanent security presence, strict time constraints may mean that adequately reviewing every release can prove a challenging task. Development methods such as Agile, in which there are many iterations in a short space of time, can often intensify this challenge.

Custom Development

Mobile applications are typically developed by either in-house developers or third-party development teams, or in some cases a combination of the two. In general, when organizations are regularly developing multiple applications, components that have been thoroughly tested will find themselves being reused across projects; this often promotes more robust and secure code. However, even when applications reuse established components from other projects, seeing libraries or frameworks bolted on to the project that may not have been developed by the project team is not uncommon. In these cases, the main project developers may not have full awareness of the code and misuse could lead to the introduction of security defects. Furthermore, in some cases the libraries may contain vulnerabilities themselves if they have not been thoroughly security tested. An example of this is the `addJavascriptInterface` vulnerability that affected the Android Webview component and when exploited resulted in a remote compromise of the device. Research found that this vulnerability was bundled with the libraries used to provide ad integration and potentially affected a significant number of applications (`https://labs.mwrinfosecurity.com/blog/2013/09/24/webview-addjavascriptinterface-remote-code-execution/`).

The OWASP Mobile Security Project

The OWASP Mobile Security Project (`https://www.owasp.org/index.php/OWASP_Mobile_Security_Project`) is an initiative created by the not-for-profit group OWASP that is well known for its work in web application security. Given the many similarities between mobile applications and web applications, OWASP is a natural fit for promoting and raising awareness of mobile security issues.

The project provides a free centralized resource that classifies mobile security risks and document development controls to reduce their impact or likelihood of exploitation. The project focuses on the application layer as opposed to the security of the mobile platform; however, risks inherent with the use of the various mobile platforms are taken into consideration.

OWASP Mobile Top Ten

Similar to the renowned OWASP Top 10, the Mobile Security Project defines an equivalent Top 10 Mobile Risks. This section of the project broadly identifies and categorizes some of the most critical risks in mobile application security. We will now loosely summarize each of the risks described in the OWASP Top 10; for a more detailed description and remedial advice, review the project page, as shown in Figure 1-2, on the OWASP wiki (`https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks`).

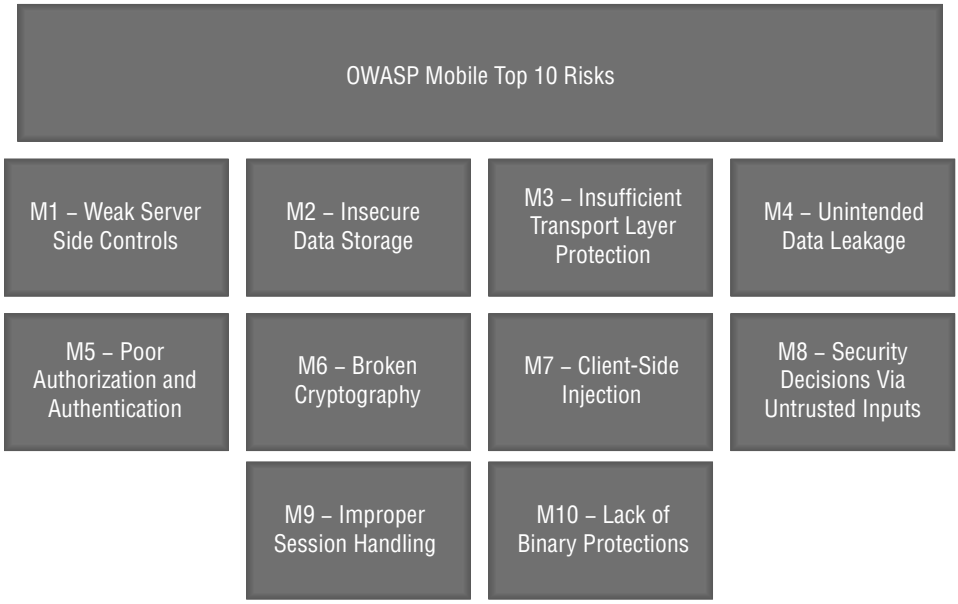

Figure 1.2: OWASP Top 10 Mobile Risks

The top 10 risks to mobile applications as defined by the OWASP Mobile Security Project are

- **M1: Weak Server-Side Controls**—This category of risk is rated as the most critical issue to affect mobile applications. The impact is rated as severe and rightly so; a serious defect in a server-side control can have significant consequences to a business. This risk encompasses any vulnerability that may occur on the server side including in mobile web services, web server configurations, and traditional web applications. The inclusion of this risk in the mobile Top 10 is somewhat controversial because it does not take place on the mobile device, and separate projects exist that explicitly cover web application risks. Although we acknowledge the severity of this risk, it is not detailed in this book because it has previously been well documented in other publications (`http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118026470.html`).

- **M2: Insecure Data Storage**—This risk relates to circumstances when an application stores sensitive data on the mobile device in either plaintext or a trivially reversible format. The impact of this risk is rated as severe and can typically lead to serious business risks such as identity theft, fraud, or reputational damage. In addition to disclosure through physical access to the device, this risk also incorporates file-system access that can be attained through malware or by otherwise compromising the device.
- **M3: Insufficient Transport Layer Protection**—This flaw pertains to the protection of network traffic and would be relevant to any situation whereby data is communicated in plaintext. It is also applicable in scenarios where traffic is encrypted but has been implemented in an insecure manner such as permitting self-signed certificates, performing insufficient validation on certificates, or using insecure cipher suites. These types of issues can typically be exploited from an adversary positioned within the local network or from within the carrier's network; physical access to the device is not required.
- **M4: Unintended Data Leakage**—This problem manifests in cases when a developer inadvertently places sensitive information or data in a location on the mobile device where it is easily accessible by other applications. More often than not this risk arises as a side effect from the underlying mobile platform and is likely to be prevalent when developers do not have intimate knowledge of how the operating system can store data. Frequently seen examples of unintended data leakage include caching, snapshots, and application logs.
- **M5: Poor Authorization and Authentication**—This category of risk relates to authentication and authorization flaws that can occur in either the mobile application or the server-side implementation. Local authentication within a mobile application is relatively common, particularly in applications that provide access to sensitive data and need to operate in an offline state. Where appropriate security controls have been missed, the possibility exists that this authentication can be bypassed to provide access to the application. This risk also pertains to authorization flaws that can occur on the server-side application and may allow a user to access or execute functionality outside the scope of her privilege level.
- **M6: Broken Cryptography**—The concept is widely accepted that applications that store data on the mobile device should encrypt it to maintain the confidentiality of the data. This risk addresses those cases where encryption has been implemented, but weaknesses exist in the implementation. In a worst-case scenario, this issue may allow an attacker to elicit portions of the plaintext or even retrieve all the original data in its unencrypted form. More often than not these risks arise from poor key management

processes such as baking a private key into the application, hard-coding a static key, or using a key that can be trivially derived from the device, such as the Android device identifier.

- **M7: Client-Side Injection**—Injection attacks can occur when a mobile application accepts input from any untrusted source; this may be internal to the mobile device such as from another application, or external, such as from a server-side component. As an example, consider a social networking application that allows many users to post updates. The mobile application retrieves other users' status updates from the site and displays them. If an attacker were able to create a malicious update that was stored on the site and then later retrieved by other mobile application users and populated into a web view or client-side database, the potential exists for an injection attack to occur.
- **M8: Security Decisions Via Untrusted Inputs**—This risk covers cases where a security decision is made based on input that has originated from a trusted source. In most cases this risk will relate to an Inter-Process Communication (IPC) mechanism. For example, consider an organization that has a suite of applications that all communicate with the same back end. The developer decides that rather than having each application prompt the user for credentials, the applications can share a single session token. To allow each of the other applications access to the session token, an IPC mechanism such as a content provider is used to share the token. If the IPC mechanism is not properly secured, any other malicious application on the device could potentially query the IPC interface to retrieve the session token and compromise the user's session.
- **M9: Improper Session Handling**—Session management is an important concept in application development; the session is the mechanism that the server side uses to maintain state over stateless protocols such as HTTP or SOAP. This risk incorporates any vulnerability that results in the session tokens being exposed to an adversary and somewhat overlaps the concepts in "A2 – Broken Authentication and Session Management" in the web application Top 10 project.
- **M10: Lack of Binary Protections**—This risk addresses the defensive protections that a developer can and in many cases should build into a mobile application. Binary protections will typically attempt to slow down an adversary that is attempting to analyze, reverse-engineer, or modify an application's binary code.

The Top 10 project is undoubtedly a useful resource for raising awareness of the types of vulnerabilities that can occur in mobile applications. As mobile application security continues to grow we expect that the top 10 project will

evolve to cover new threats as they are discovered, and play an even more important role in educating developers and security professionals.

OWASP Mobile Security Tools

Whether their purpose is for simply supplementing manual assessments, providing a framework for the development of other tools, or as a resource to offer remedial or hardening advice for developers, tools are an important part of any security professional's arsenal. The OWASP Mobile Security Project has developed a number of open source security tools (`https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Mobile_Tools`) for the community that you may find useful in your learning. We briefly describe each of them now:

- **iMAS** (`https://www.owasp.org/index.php/OWASP_iMAS_iOS_Mobile_Application_Security_Project`)—Created by the MITRE Corporation, this project is an open source secure application framework for iOS. It provides an ideal resource for developers or security professionals who want to learn or understand how to implement security controls for the iOS platform. The goal of the project is to demonstrate and provide implementations protecting iOS applications and data beyond the Apple-provided security model and as a consequence reduce an adversary's ability to reverse engineer, manipulate, and exploit an application. To achieve this goal, the project has created a number of open source implementations that address several areas of common vulnerability, including in-application passcodes, jailbreak detection, debugging protection, and runtime validation. Although we delve into some of these topics in great detail in Chapters 2 and 3, the iMAS project is certainly a useful resource for learning defensive techniques or as a reference for developers.
- **GoatDroid** (`https://www.owasp.org/index.php/Projects/OWASP_GoatDroid_Project`)—The GoatDroid project developed by Jack Mannino and Ken Johnson is a self-contained training environment for Android applications. The environment provides two sample implementations to hone your skills: FourGoats, a location-based social network, and Herd Financial, a fictional mobile banking application. Between them, these two projects provide broad coverage for most of the OWASP Top 10 Mobile Risks and are a good starting point for beginners in Android application security.
- **iGoat** (`https://www.owasp.org/index.php/OWASP_iGoat_Project`)—Similar to the GoatDroid project, iGoat is a training application for improving your iOS assessment knowledge. The project is developed by Ken van Wyk, Jonathan Carter, and Sean Eidermiller and is open source

(`https://code.google.com/p/owasp-igoat/`). It provides both a server and client application with a number of exercises covering important topics such as local storage, the key chain, SQL injection, and more.

- **Damn Vulnerable iOS** (`https://www.owasp.org/index.php/OWASP_DVIA`)—This project, created by Prateek Gianchandani, provides another vulnerable iOS application for training purposes. In conjunction with the iGoat project, the two applications provide good coverage of the OWASP Top 10 Mobile Risks. The application is comprised of several challenges that you can complete to further your understanding, including topics that are omitted from iGoat such as jailbreak detection, runtime manipulation, patching, and cryptography.
- **MobiSec** (`https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_MobiSec`)—MobiSec is a live environment for penetration testing mobile applications; it is created by Tony DeLaGrange and Kevin Johnson. The idea behind the project is to provide a single resource to host and maintain the latest versions of all the individual tools you might need during a mobile application assessment, in a similar way to other live distributions such as the popular Kali Linux, but in this case specifically focused on mobile security.
- **Androick** (`https://www.owasp.org/index.php/Projects/OWASP_Androick_Project`)—This project addresses a slightly different topic from the other projects and is focused on automating forensic analysis tasks for Android applications rather than penetration testing or self-learning. The project, created by Florian Pradines, automates the retrieval of key forensic artifacts such as APKs, application data, databases, and logs from the device.

Of course, you will encounter and even require many other tools during your adventures in mobile application security and we document many of these in later chapters. However, the OWASP projects are particularly useful for self-learning as they're well documented, open source, and specifically developed to provide coverage for the Top 10 Mobile Risks project, so we certainly recommend them as a starting point for beginners.

The Future of Mobile Application Security

The explosive rate at which smartphones and mobile applications have been adopted over the past five years has shown no signs of diminishing, and we expect this trend to continue in the future. The consequence of the growing mobile revolution will only place further emphasis on understanding the security threats that mobile deployments face as well as effective ways of addressing them. We do not believe the current threats to mobile security are at present well understood, particularly in the development communities.

As such, we expect that classic vulnerabilities such as insecure data storage and insufficient transport security will continue to be prevalent for the immediate future.

That said, mobile application security is a continually evolving landscape and we fully expect new categories of attacks to arise following advances in mobile technologies. The introduction of new hardware components such as fingerprint sensors and increased adoption in existing technologies such as NFC will undoubtedly lead to the discovery of new vulnerabilities, particularly when deployed into environments such as mobile payment processing, as used by Google Wallet and Apple Pay.

As with other areas of software and particularly those that are used to facilitate monetary transactions, criminals will seek to take advantage of vulnerabilities for financial gain. We have already seen an increase in banking malware and premium-rate SMS fraud and expect this trend to continue. This increase has already somewhat altered the threat landscape and in response, some application developers have begun to employ binary protections to defend against these threats. As awareness of these threats matures, the adoption of such protections will likely increase in prominence, along with the use of technologies such as two-factor authentication.

It is also likely that the evolution of cross-platform mobile applications will continue as developers aim to reduce fragmentation across the various mobile platforms. This has been witnessed in the growth of two development trends:

- **Browser-based applications**—This term describes applications that are usually a "mobile friendly" clone of the main site and loaded via the device's browser.
- **Hybrid applications**—This term refers to mobile applications that are a native wrapper for a webview and often use a framework to access native device functionality.

To complement these trends a large number of both commercial and freely available frameworks have been created, each with its own quirks and intricacies that can lead to a variety of different vulnerabilities. As with most changes in technology, these trends have brought with them new attacks and variations on existing attacks; we examine the security implications of these and similar ones in Chapter 18.

Despite all the changes in mobile applications no signs exist that the classic attacks are diminishing. A positive step toward addressing this, however, is raising awareness of mobile security threats and vulnerabilities through documentation, classification, and demonstrations such as those being developed by OWASP. Through this and similar projects we believe that awareness of mobile security can mature and help to provide development controls to reduce the number of mobile application vulnerabilities.

Summary

Over the past five years the increased popularity of the modern smartphone has contributed to a surge in third-party application development. Enhancements in smartphone hardware have helped applications rapidly evolve from simple standalone applications to feature rich offerings that can integrate into multiple online technologies. During this evolution several technical, economic, and development-related features have contributed to bring about a weak security posture demonstrated by many of today's mobile applications.

In addition to the traditional input-based security problems that can affect all types of applications, mobile applications are also affected by several relatively unique vulnerabilities due to the nature in which they are used. These issues are often not well understood by developers and can lead to attacks when a device is used on an untrusted network, when a device is lost or stolen, or even from other components on the mobile platform.

Research on the current state of mobile security has shown that application vulnerabilities are not well understood and that the majority of applications are vulnerable to attack. Furthermore, the evolution of new technologies and integrations is likely to produce entirely new attacks, which could pose a serious threat to organizations that do not react and adapt accordingly.