Threat Model for Mobile Applications
Security & Privacy
www.DataTheorem.com

Overview

Mobile applications (and the devices they run on) are powerful, as they can play music, check email, read documents, purchase products, get directions, play games, watch movies, scan barcodes, and so on. Due to its multiple dimensions and compact size, the threat landscape differs for mobile apps as compared with web or client/server apps. This difference is driven by the fact that mobile apps have access to certain data types never available on web apps, such as call history, SMS logs, contact lists, geo-location, etc. Thus, due to different (and more sensitive) data exposures, developers need to build mobile applications with these new threats in mind. This paper is a brief overview of the new and changing threat model for mobile applications, and their emerging attack surfaces.

Mobile Applications Threats

There are five mobile application threats discussed in this paper, including the following items:

1. Data loss from security vulnerabilities
2. Unauthorized/private data collection
3. Data exposure to other applications
4. Data exposures at-rest (on the device)
5. Data exposures in-transit (over the network)

It should be noted that the previous list is not exhaustive, as many new threats can appear rapidly. Furthermore, mobile malware is grouped under items #1, #3, #4 and #5.

Attack Surfaces on Mobile Apps

The attack surface, which is the part of an application that is usually targeted for security exploits, of a mobile application can be comprised of several technologies, depending on the implementation. For example, some mobile applications are simply mobile websites, using traditional HTML/Javascript technologies. On the flip side, other mobile applications are native clients offering a full list of rich features built-in by the iOS, Android, or Windows Phone operating systems. Lastly, there are mobile applications that are a blend of the two, using a significant native experience, but also leverage mobile web technologies (especially for screens that must have a consistent experience between iOS, Android, and Windows apps). Also, in order to support both native apps and mobile web, there must be some use of server side web services and APIs (heavier for native clients and lighter for mobile web). Below is a short list of the possible technologies (and attack surfaces) that are leveraged by mobile apps:

- Compile Binary
  - .ipa: Objective C
  - .apk: Java
  - .xap: C#
- Server Side Components
  - Web Services
  - APIs
- Mobile Web
  - HTML/Javascript Screens
- Network Communication
  - HTTP/HTTPS

Threat Model for Mobile Applications Security & Privacy

www.DataTheorem.com

Threat 1: Security Vulnerabilities

Mobile applications are susceptible to security vulnerabilities from code level issues and/or run-time flaws in the application. Similar to any desktop or web application, code-level security issues in Objective-C (iOS), Java (Android), and C# (Windows Phone) can harm the application, its users, and the data it controls. It should be noted that all three mobile operating systems leverage a sandbox model, which separates one app from another, unlike desktop apps or web applications. While the sandbox is not foolproof (see threat #3), it does reduce the likelihood of a code level security issue from being externally exploitable (rather than self-contained within the sandbox). Similar to any other application, secure coding techniques should be leveraged in order to reduce the attack surface for any possible data exposure.

In addition to code level issues, another avenue for security vulnerabilities is application logic attacks, where attackers abuse the application using existing (or the lack thereof) security controls. Common run-time attacks include, but are not limited to, escalation of privilege, authentication bypass, and session manipulation. The idea of hunting for run-time security issues in a mobile application has a similar mindset from the mobile application world: Use the app, identify flaws/exposures, and exploit these flaws/exposure for unauthorized access to data.

Threat 2: Unauthorized/Private Data Collection

Mobile applications can be guilty of pulling data from a device/user without explicit permission, or worse, pulling data that is considered private and should never leave the device itself (even over SSL). As we all know, mobile devices have far more data about their owners than device counterparts such as PCs, Macs, or web browsers. The list is long and grows every day, as mobile devices have information on an individual's exact whereabouts, names, emails, home addresses, phone numbers, call history, call frequency, SMS data, professional and personal schedules, stock portfolio/interests, online browsing habits, etc. Furthermore, if the data downloaded by applications is included as well, this list goes on to include health information, eating habits, exercise routines, credit card numbers/images, date of birth, checking account numbers, check images, and so on. The threat here is collecting private data without the user's consent or knowledge. There is a fine line, as many features within mobile applications leverage the above data, but if taken without permission, without a valid business need, or stored for long periods of time, possible fines can be expected (as recently seen by the California Attorney General's office - see more information here: Privacy on the Go). In addition to unauthorized data collection, there is some data that is private and should never be collected by the mobile application (as they are considered tracking IDs linked to specific individuals). For example, hardware identifiers (e.g. UDID or IMEI) are glorified tracking IDs, and should be avoided at all costs. These items were so sensitive, it prompted Apple to deprecate UDID (hardware identifier) in favor of IDFA and IDFV.

Threat 3: Data Exposure to Other Applications

Mobile applications may directly or inadvertently expose their data to any other application running on the system. iOS, Android, and the Windows Phone operating systems have a sandbox model to prevent data sharing between one application and another. The sandbox is designed to not only keep a user's data safe, but to keep applications separate from each other. While the implementation of the respective sandboxes differs (e.g. in Android, each application runs under its own UID, but this is not the case in iOS), they are all designed to create silos for each application. The threat arises when the sandboxes do not isolate one application from another, either by design or implementation flaw. Mobile developers must be aware of known and legitimate data sharing capabilities in each platform, such as the UI Pasteboard (iOS), SD Cards (Android, Windows Phone), temp directories, cache locations, log files, etc. The list is not only fairly long; it grows every day with each new release of iOS, Android, and Windows Phone. Furthermore, application data stored in areas that are considered private might actually have some level of exposure to other apps on the device. For example, if a phone has been jail-broken or rooted, all data stored by an application is accessible to all of the other apps by default. Further, if there is a security issue in the platform, it may allow one app to access the private directory of another app (see http://securityshastra.blogspot.com/2013/07/remote-code-execution-in-android.html). This list can go on forever; therefore it is important to ensure data sharing between applications is studied closely.

Threat 4: Data Exposure At -Rest

As online activity continues to migrate from traditional web applications to mobile apps, the exposure of a user's data simply increases. While users think of mobile applications in the same way as web applications, the architecture is closer to a client/server model (where web applications are closer to the mainframe/green-screen model). In the client/server and native mobile app model, data is often stored locally on the device, both during active sessions or even after logoff. The type of data stored on mobile devices ranges from highly sensitive (medical data) to informational (tweets). Unlike data within web applications (where data is mostly server side), sensitive, confidential, and private data is now with you everywhere you go. While this was the same for client/server apps running on desktops or laptops, the key difference is that your mobile device is guaranteed to be lost, stolen, traded-in, upgraded, gifted, exchanged, or simply re-sold, probably within two or three years after purchase. Hence, what type of data can you store on a device that will eventually be in the control of unauthorized parties? There is no easy answer to this question and it is vastly different from one application to another (and one organization to another); however, it must be scrutinized closely to ensure the exposure remains minimal. In terms of the threat model, it is important to know what data your application has left for others to see, use, compromise, or exploit when a remote attack or physical compromise has occurred. For example, usernames, passwords, password equivalents, PII, PHI, or PFI are items that should not be stored locally and wiped off the device after use.

Threat 5: Data Exposures In -Transit

Mobile applications are highly exposed to data in-transit threats, which is a well-known security & privacy attack class. Due to the high likelihood of mobile devices joining non-trusted networks, malicious actors can target the communication layer much easier than before (as compared to desktops or laptops). For example, the in-transit threat during the PC era was much smaller than it is today. The communication from your office desktop to local servers was a threat, but did not expose much to external attackers. During the laptop era, the threat increased as users began to leverage their PCs at work, at home, and a few places in between. While the few places in between grow daily, the dominant locations still remain home and work. Now comes the mobile era, where it should be assumed your mobile device will join a hostile, unknown, or rogue network. For example, not only will the words "Free Wi-Fi" encourage the masses to join unknown and hostile networks, but the fact that mobile carriers limit data plans will drive users towards public Wi-Fi networks to save precious bandwidth quotas. Thus, a mobile application must assume the network is not trusted, and actively build defenses against network level attacks (e.g. downgrade attacks on SSL) or clear-text communication (e.g. Geo-location sent in clear-text, for all attackers to snoop). By actively assuming attacks on SSL and clear-text protocols taking place, mobile application defenses can be installed to keep the app, its data, and its users safe even in the most hostile network, which is guaranteed to be joined by thousands of users over an app's lifetime.

Threats and Examples

Now that we have briefly discussed five mobile application threats, let's dive deeper into some specific examples. The following section lists specific attacks that fall under each respective threat. It should be noted that this list is non-exhaustive, but just a brief example of each threat class:

- Threat 1: Security Vulnerabilities
  - Objective-C/Java (Code level) security issues
  - Local or remote injection
  - Application logic issue (e.g. Auth bypass)
- Threat 2: Unauthorized/Private Data Collection
  - Retrieval of UDID or IMEI
  - Bundled Geo-location, IDFA, and UserID
  - Collection of contact list/pictures/SMS logs
- Threat 3: Data Exposure to Other Applications
  - Passwords shared via UIPasteboard
  - Data stored in SD cards or temp directories
  - Data caching to public locations
- Threat 4: Data At-Rest Exposures
  - Password persistent on file system
  - Browsing history stored in temp directories
  - PII or financial data stored on the device
- Threat 5: Data In-Transit Exposures
  - Geo-Location leaked to 3rd party source
  - SSL Not Fully Validated
  - Credit Card numbers transferred in the clear

Threats vs. Risk

While mobile security threats are wider than before, it is up to each organization to decide the level of risk. For example, let's say a mobile application has persistently stored an authentication token on a user's device. A mobile banking application would probably deem this as high risk, as anyone who gains control of the device would have full access to the user's bank account and assets. On the flip side, a news reader application might classify it as a low risk since it only exposes publically accessible new articles. While the security issue is the same (password-equivalent values stored persistently on the device), the risk of the issue differs from one application to another.

The one caveat to an organization-driven risk model is outside regulations and standards. Organizations must make their own risk-based decisions, but they should adhere to industry standards and regulations for both security and privacy. Using the previous example, it is never best practice to store passwords or password equivalent values (e.g. authentication tokens) at-rest on a device (either PC, Mac, or mobile). This behavior would be deemed atypical for most security professionals. Furthermore, privacy advocates may consider a user's reading habits as private data. Hence, while the articles a person reads are public on the Internet, the fact that a user's reading habits are now exposed to unauthorized actors might be a privacy violation. In either case, it is important that organizations make their own risk decisions as they pertain to mobile application threats, and adhere to all security and privacy standards as well.

Conclusion

The threat model for mobile application security is different than previous application platforms. For example, Geo-location leaked to 3rd party destinations, applications accessing phone call logs, or passwords stored on devices that were left behind at the Gourmet Haus Staudt restaurant are all issues that developers never had to face, **but must address now**. Further, adding existing security threats, such as code-level security flaws and application logic issues, not only makes the challenge wider, but forces developers to face both security threats and emerging privacy issues as well. While things may seem grim, the bright side is all three major operating systems, including iOS, Android, and Windows Phone, are designed better than a generation ago. It is clear that a decade's worth of security knowledge has been used in designing these new mobile operating systems, drastically reducing (and even removing) some attack surfaces that were so prevalent before. Hence, while it is clear security is moving forward with mobile OSes, many new threat classes have appeared as well. These threats must be accounted for when trying to keep data secure & private on mobile applications.

For more information, please contact:

- Data Theorem, Inc.
- http://www.DataTheorem.com
- Email: info@datatheorem.com
- Phone: 415.763.7331