© West1 | Dreamstime.com

Secure Mobile Application Development

Jeffery Payne, *Coveros*

Mobile applications are a perfect storm for organizations seeking to ensure the security of their commercial products and IT services. Not only are current Web security challenges relevant in the mobile world, but so are the traditional "fat client" security concerns from decades ago. To make matters worse, each mobile platform has nuances that reduce the effectiveness of certain security controls and make it difficult to attain cross-platform application security.

Mobile Security Challenges

Mobile security challenges stem from change in the *threat model* associated with the products and services providing mobile applications and interfaces. A threat model is a depiction of a system's attack surface, annotated with possible threats and the ways in which critical assets might be targeted. Threat modeling is the process of analyzing threat information, determining which attack vectors a threat might follow to compromise a system, and putting in place appropriate security controls to protect critical assets.

The relevant security risks and concerns depend on the architecture of the mobile application. For example, a mobile application that only provides the front end to an organization's website will have different security concerns and challenges from an online banking application with a fat client that deals with sensitive financial data.

In general, mobile applications have a different threat model from traditional Web applications.

Changing Attack Profiles

Because of their wide accessibility, both Web and mobile applications face attacks from a variety of directions: malicious mobile users, third-party applications, and users seeking to directly access back-end systems. However, with mobile applications, such attacks have a greater chance of succeeding.

Malicious users. Mobile devices are often lost or stolen, providing malicious users greater accessibility to private user data and critical application credentials. Mobile applications that don't properly manage sessions or that provide local mechanisms for remembering user IDs and passwords are easily compromised. For example, sessions are often left open on mobile applications for long periods of time so mobile users can seamlessly pick up where they left off when bringing an application to the foreground. Not closing open sessions on a regular basis increases the likelihood that a malicious user can gain unauthorized access to critical data and applications.

Furthermore, relying solely on device security, such as auto-lock and password or PIN protection is insufficient, because these mechanisms are easily bypassed once the device is in hand. A malicious user can perform "rooting" or "jailbreaking" on most mobile devices to circumvent system-level security mechanisms.

Malicious third-party applications. Mobile platforms are much more susceptible to malicious applications owing to the practice of downloading and running mobile apps acquired from mobile app stores. Best practices in Web security often prohibit the downloading and executing of any content coming from an untrusted third party. Unfortunately, this model won't work for applications that must reside on the mobile device.

To secure these applications, mobile device platforms place them in an *application sandbox*, which constrains their ability to interact with other applications or access unauthorized resource. Each platform has different mechanisms for letting mobile applications communicate and share data when needed. Security breaches are possible if these mechanisms aren't used properly, if application configuration or permissions aren't set up correctly, or if your application doesn't secure its data.

Malicious access to back-end systems. A common attack in Web applications is to circumvent the front end and attempt to gain authorized access to a back-end system. Mobile applications are susceptible to these same types of attacks, but they often provide attackers with additional back-end system information, helping them breach security.

For example, a malicious user might purchase your online banking application, reverse engineer the code, and discover the location and permissions of administrative login pages and forms. Such information might be useful for attempting to directly access administrative login pages and circumvent security.

Fluid Trust Boundaries

Trust boundaries are the delineations between various levels of trust within a system architecture (untrusted user, regular user, application administrator, and so on). Part of the threat-modeling process is to identify trust boundaries and the information flowing across them. Understanding trust boundaries lets developers properly protect this critical information and validate users in various parts of the system. Secure development principles state that users shouldn't be given greater trust than necessary to perform their tasks. Mobile applications often have more fluid trust boundaries than other systems for a variety of reasons.

Trusting untrusted applications. Mobile devices blur trust boundaries, because trusted and untrusted applications live on the same device and often must interact. From a security perspective, third-party applications must be considered untrustworthy; however, some degree of trust must typically be given to these applications for them to work as advertised. For example, an online game might request access to SMS services so it can send you notifications during a game. This might seem like a reasonable request, but if the application is malicious, it could easily misuse this capability to flood your phone with messages.

Even though there are significant challenges to building secure mobile applications, there are ways to mitigate risks.

Accessing sensitive data. Unlike mobile Web applications, which maintain as much sensitive data on the back end as possible, mobile application clients often need to operate using sensitive data on the mobile platform. If a mobile application isn't carefully designed, sensitive information could be compromised or an unauthorized user might be able to perform transactions.

Mobile platforms use different approaches to protect locally stored application data. Regardless of the approach, local data isn't always physically deleted when an application performs a delete operation. Mobile devices typically use flash memory (such as NAND memory) for local storage, because it can be quickly accessed. Because there's a limit to the number of times NAND memory blocks can be erased before they become inoperable, mobile devices don't regularly physically delete data. This provides malicious users and applications the opportunity to access sensitive information that the application developer thought was no longer present on the device.

Mobile Platform Nuances

Every mobile platform has its own approach to file management, memory management, application configuration, application security, networking, and so on. This makes it more difficult not only to secure your applications but also to secure them across a variety of platforms.

For example, Andriod uses Linux-style file permissions, and each application can't access another application's files unless given explicit permission to do so. However, Android also supports the ability to read and write files to and from an external storage device. By default, files written to such a device have global read and write permissions and will be accessible by any application that knows where to look. Not understanding this security nuance could result in unauthorized access to sensitive data.

Building Secure Mobile Applications

Even though there are significant challenges to building secure mobile applications, there are ways to mitigate these risks. Here, I discuss a few of the best practices associated with secure mobile development. A list of the Top 10 Mobile Risks is maintained by the Open Web Application Security Project (OWASP) and contains additional information on how developers can protect their mobile applications from attack—see https://www.owasp.org/index.php/OWASP_Mobile_Security_Project. This information also includes specific advice when building applications for iOS (Apple) and Android (Google).

Keep all sensitive information on the system's back end, and securely transmit and display only what you need.

Don't Store Sensitive Data Locally

Although it might complicate your overall architecture, too much risk is associated with storing sensitive data on a mobile device. Keep all sensitive information on the system's back end, and securely transmit and display only what you need and only for as long as you need it. Don't save any of this information to a file, because deleting it doesn't mean others can no longer access it.

Also, many mobile devices capture screenshots of applications when they're moved to the background (allowing faster screen refresh when the application is brought to the front again), so it's best to show as little sensitive information on the screen as possible. Captured screen shots will persist after deletion and could reveal sensitive information to those who access them. Turn off all caching of information when handling sensitive data to ensure such data isn't left in memory.

If your application must store sensitive information locally, use the strongest encryption libraries shipped with your mobile device, along with a master key (or key chain) encrypted with a user passphrase using a key derivation function. Passphrases are often short, so this approach might be susceptible to a brute force attack unless you select a key derivation function that iterates enough to consume a significant amount of CPU time. If your system has a back end, maintain the master key there. Don't use external storage for any sensitive data. Make sure permissions are set such that only your application can access its local storage.

Close Down Idle Sessions

Mobile applications typically leave sessions active longer than Web applications do, increasing the risk of nefarious behavior by unauthorized users or applications. This is done as a convenience to mobile users, so they don't have to reauthenticate to continually use an application. The longer a session is active, the more likely an attacker can perform a session hijack—an approach to stealing a session ID and becoming a legitimate user—so session times should be limited. In general, no idle session should be allowed to go longer than five minutes before it shuts down.

Don't Trust Clients

Due to the enhanced threat of malicious users and malicious third-party applications on mobile devices, server-side logic should never assume that a client application is legitimate. Use common server-side security practices in a mobile application environment, just as you would if interacting with an unknown user on the Web. Some suggestions include the following:

- disable lower levels of encryption (export grade) so no application is permitted to communicate with a server using a less secure transport mechanism,
- validate all input received from the client,
- disable verbose errors and messages,
- return the minimum server response at all times,
- change all default directories for where information is maintained, and
- give a standard response to invalid user name or password requests.

Don't Forget about Native Code

Most mobile platforms support the creation of native code applications, letting code be written in languages that are vulnerable to traditional attacks, such as stack buffer overflows, memory corruptions, heap overflows, and race conditions. Fortunately, mobile platforms today support address space layout randomization (ASLR), which randomizes where various types of information are kept. Because overflow and memory corruption vulnerabilities often need to know the ordering of system information to perform their attack, ASLR greatly reduces the effectiveness of these particular types of vulnerabilities. Make sure your applications are built with ASLR enabled.

Beyond ASLR, other traditional vulnerabilities for a given language are potentially still exploitable. Code revenue and code scanning should be used to identify code-level vulnerabilities and eradicate them before release.

Understand Your Platform

Your mobile application will only be as secure as your understanding of how the mobile platforms you deploy operate. Before building your application, you need to understand what vulnerabilities have been identified for your platform in the past and make sure you have the latest version of the operating system installed.

You should also learn how application data is stored, how it's protected from access, and when it's physically deleted from the device. Some mobile platforms now provide functionality for applications to physically delete sensitive data when necessary. Be aware that doing so will render blocks of storage unusable over time.

You also need to understand the default configurations for applications, the mobile browser, and application communication permissions so you can properly protect your application.

Furthermore, you should learn how and when information is cached, keyboard keys are logged, and screenshots are saved. In general, you'll want to disable all of these capabilities when operating on sensitive information

Finally, you need to understand how libraries your application uses are loaded and run. Statically link your applications at compile time to avoid the possibility of a malicious user or application replacing a legitimate library with a malicious one loaded at run time. IT

References

1. Apple, *iOS Security*, May 2012, http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf.
2. W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," *IEEE Security & Privacy*, Jan./Feb. 2009, pp. 50–57.

***Jeffery Payne** is CEO of Coveros, a software firm that specializes in secure software development. His research interests include mobile application security, malicious code analysis, continuous integration, automated testing, agile development methods, and secure software development. Contact him at jeff.payne@coveros.com.*