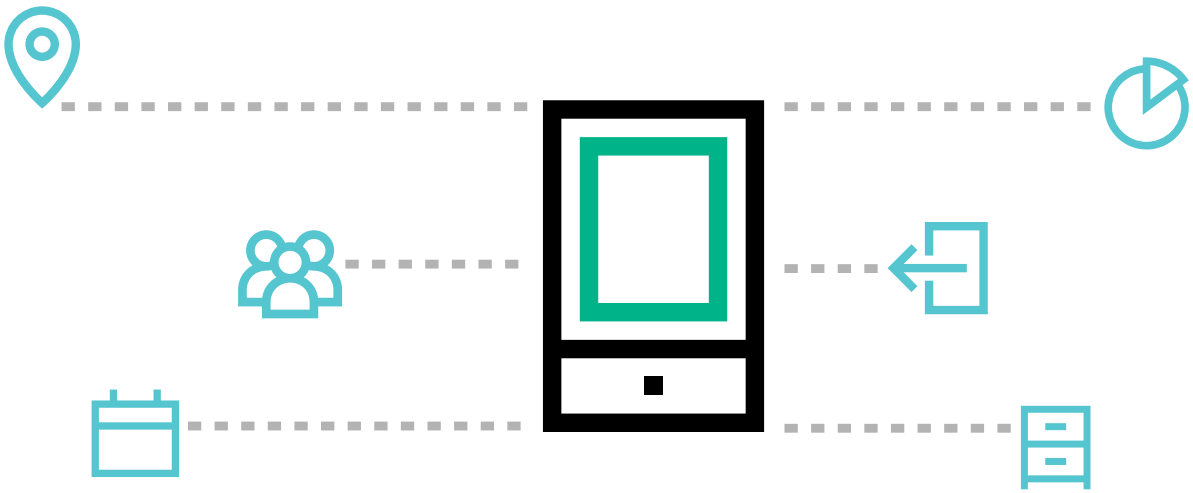
Hewlett Packard Enterprise



Mobile Application Security Report 2016

Overview

HPE's first mobile application security study was published in the fall of 2013. At that time we analyzed 2,000 applications and found that 97% of them accessed at least one private information source. Now, over two years later, and after testing over 36,000 applications, the situation has not improved; 96.52% of applications were flagged in at least one of the 10 core privacy checks.

Mobile applications collect a lot of data.
From navigation to banking, mobile applications have quickly transformed modern life. However, these applications often collect unnecessary data, which is concerning for a number of reasons. In our analysis of more than 36,000 Android and iOS applications, we found that personal information such as contacts, calendar data, and geolocation information were broadly accessed beyond those applications that actually need that data to operate. While the data collected is seemingly innocuous, the potential threat to privacy and reputation is very real. So we asked ourselves: *What is the impact of increasing data collection? Are mobile application developers equipped to protect the data we provide? How can corporations best protect themselves from these threats?*

Testing Methodology

Between May and November 2015, Fortify on Demand performed binary analysis of more than 36,000 unique applications downloaded from the Apple App Store and Google Play. The applications spanned 27 categories.

Categories Tested

Books	Health & Fitness	Photography & Video
Business	Libraries & Demo	Productivity
Comics	Lifestyle	Reference
Communication	Media & Video	Shopping & Catalogs
Education	Medical	Social Networking
Entertainment	Music & Audio	Sports
Finance	Navigation & Transportation	Tools & Utilities
Food & Drink	News & Magazines	Travel & Local
Games	Personalization	Weather

In cases where multiple versions of the same application were downloaded, we counted it one time only; in other words, a vulnerability in any version of the application during the designated time period is counted only once.

Report Findings

Data Collection and Privacy Violations

52.1% of applications accessed geolocation data

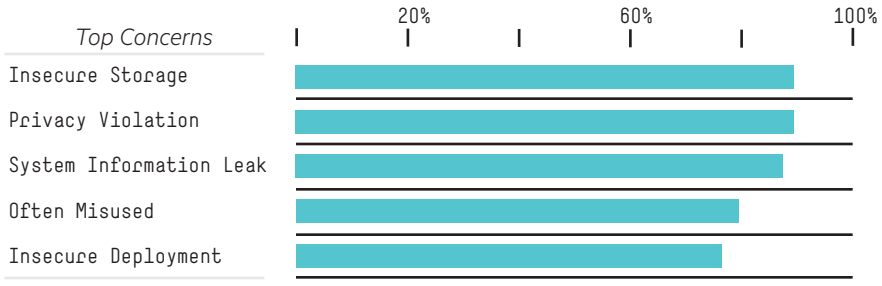
The Ashley Madison breach in 2015 is one of the more widely publicized examples of data collection gone awry.

The risks of collecting geolocation data may not be immediately apparent, but recent headlines demonstrate how it can be disastrous. The Ashley Madison breach in 2015 is one of the more widely-publicized examples of data collection gone awry – their storage of geolocation data allowed a reporter to pinpoint the location of otherwise anonymous users (http://globalnews.ca/news/2471012/what-a-map-we-cant-show-you-tells-you-about-your-phones-location-settings/). Additionally, in the age of highly targeted attacks, it's possible that geolocation information could be used to target an individual in the real world...with all the potential dire consequences that implies. What's more, the study found that more than 70% of education applications on iOS accessed geolocation data – applications often marketed towards children.

11.5% of applications accessed contacts, while

16.3% of applications accessed calendar data

The collection and leakage of personal data on mobile devices isn't just a risk to users, but for their employers as well. When applications collect information about a user's social network, chances are good that professional contacts are included in the mix. Calendar data can be even more sensitive, detailing not just when meetings take place but also the topics and invitees. Our analysis found that the applications accessing this data weren't necessarily what you'd expect. For instance, we weren't surprised to see 40.9% of social networking applications accessing contacts, but did not expect to see 19.8% of the finance applications we reviewed doing the same. Additionally, calendar data was accessed by 41.9% of the iOS games and 52% of the iOS weather apps we reviewed.



Insecure storage and privacy violation weaknesses are two of the top five concerns cited in the HPE 2016 Cyber Risk Report.

A framework that is misconfigured—or insecure to begin with—could be storing or transmitting a significant amount of highly specific and potentially sensitive data about users.

How does data get exposed?

61.7% of applications used ad or analytics frameworks

Ad and analytics frameworks are commonplace in app development, but developers must take extra care when sensitive user data is involved. We found that 64.8% of health and fitness applications and 53.2% of medical applications we reviewed used ad or analytics libraries, as did 43.8% of the finance applications we tested. A framework that is misconfigured – or insecure to begin with – could be storing or transmitting a significant amount of highly specific and potentially sensitive data about users. Areas where data is commonly exposed by third-party ad and analytics frameworks include caches and temporary files on the device, system logs, URLs, and unencrypted transmissions to backend servers.

What can developers do?

- Ensure that frameworks are configured for production, if applicable, and not using debugging modes that may record more data on the device.
- Verify through testing that frameworks send traffic over HTTPS only, and do not transmit sensitive data in the URL.
- Check the device and application sandbox for files written by third-party frameworks, and review each for sensitive data.
- Collect only the minimum amount of data necessary.

94.8% of applications include logging methods

Unnecessary logging can expose data to unauthorized third parties.

During application development, logging can be a critical component of correcting buggy code. But once an application is running on a user's device, unnecessary logging can expose data to unauthorized third parties. Don't assume code is free of logging simply because you didn't include it yourself. If an application includes third-party code, logging may still be part of the package. Of particular concern is how an application behaves when unexpected errors occur: are stack traces or detailed crash information output to the system log? If so, the application's internals may not be all that is on display. Error messages may also include user account credentials, personally identifiable information, or other sensitive data.

What can developers do?

- Use macros to remove logging statements from production builds.
- Frequently review the system log to understand what third-party code may be logging.
- Avoid printing stack traces or dumping the contents of data structures when errors occur.

70.6% of applications can access external storage

Applications cannot trust that files read from external storage have not been modified by a third party.

Applications built for Android have the option of storing data to external storage, such as on an SD card. While convenient, external storage is not intended for use with sensitive or high-integrity data, particularly since versions of Android prior to 4.4 (around 30.6% of devices as of January 2016, developer.android.com/about/dashboards/index.html) lack the ability to enforce app-specific permissions for files written to external storage. This means that applications cannot trust that files read from external storage have not been modified by a third party; similarly, applications should not trust that data written to external storage can't be accessed by third parties. Despite this, 65.9% of finance applications that we tested requested external storage permissions, as did 56.1% of the medical applications we reviewed. Given the sensitivity of the data these types of applications store, we hope they are only storing low-integrity, non-sensitive data – or securely encrypting data prior to storing it.

What can developers do?

- Avoid storing sensitive data on external storage. If sensitive data must be stored on external storage, ensure it is securely encrypted.
- Never store executable code, class files, or files that will be loaded in a WebView on external storage unless they are cryptographically signed, and verified prior to load.
- Follow input validation best practices when loading or parsing data from external storage. Treat all data originating from external storage as untrusted.

Conclusion

The benefits of mobile applications are undeniable. However, new technology also invariably introduces unintended security consequences. For mobile devices, a large part of those consequences reside in how applications collect, store, and communicate sensitive data. The responsibility rests on the enterprise to empower and entrust developers to treat data collection with the utmost respect and caution. And, consumers must also be responsible and aware of what is being collected and the potential consequences of it.

Recommendations

Hewlett Packard Enterprise has the following recommendations for those looking to maintain their own privacy or that of their customers.

Best Practices for the Enterprise

- **Leverage static analysis tools.**
  For mobile applications with available source code, use a static analyzer to identify code paths where data could be leaked. This includes code that writes data unencrypted to the file system, includes it in log messages, or sends it unencrypted over the network.
- **Analyze apps during and after use.**
  Mobile applications aren't static, so don't rely solely on static analysis to find data leakage. Step into the shoes of the target user and use the app as they would – logging in, granting requested permissions, and exercising all the functionality. Then analyze the files generated by the application at key points – such as after logging in or out, or after using functionality that requires access to sensitive data. For maximum granularity, use a tool that monitors filesystem access at runtime, and keeps a log for review. Check file types before you perform analysis in order to detect and handle compressed files and encoded data. Simple string searches can yield quick results for many key artifacts:

– System logs

– Application's designated data directories

– External storage

– Application cache files and binary data files

- **Review interactions with third parties.**
  In addition to analyzing in-house code, it is important to understand the impact of third-party code on your application. Review the application's linked frameworks and libraries by using a tool like otool for iOS applications, and by decompiling the application package for Android. Check for known vulnerabilities in third-party code by consulting a CVE database in addition to checking the third-party provider's own website. If libraries are compiled in that send data to third-party servers, check that the scope of data transmitted is accounted for in the application's privacy policy.

  Even if third-party code isn't used in an application, it could still communicate with other processes on a device to initiate actions within the app, or transmit data to other apps. Look for sharing mechanisms like content providers, sharing via the pasteboard, and test whether an unauthorized application on the device can obtain data via those mechanisms. Check whether sensitive actions can be initiated by an unauthorized application via intents or URL schemes.

- **Don't just trust SSL, verify!**
  Transmitting data over SSL might seem like a silver bullet, but don't assume that applications are implementing it correctly. Try to proxy the traffic from the target application and observe whether any unencrypted traffic is transmitted – even when in-house code travels over SSL, you may find that third-party frameworks transmit in the clear. Further, if you are able to capture HTTPS traffic in your proxy, it is likely that the application is not performing validation correctly, if at all. Install your proxy's CA certificate on your device to test whether the application is employing certificate pinning – all applications should, if feasible.

  Once you are capturing traffic, you'll also have an opportunity to inspect the data being transmitted. Watch for sensitive data being transmitted over unencrypted channels, data transmitted to third parties, and sensitive data embedded in URLs.

- **Root out stored secrets.**
  Perhaps the application you are testing is doing everything right – it even encrypts data prior to storing it on the device! Sadly, even the strongest encryption offers no protection if the key is stored in plain sight. Look for strings in the binary executables and decompiled source files that could indicate hard-coded encryption keys, passwords, and initialization vectors. Review configuration files included in the application package – usually plist files on iOS, and XML on Android – for hard-coded secrets. Source code and even header files dumped from the executable may help you identify what the application uses for encryption operations, and lead back to stored secrets on the device. And don't forget to examine the ciphertext itself to ensure it isn't just trivially encoded with an algorithm like Base64.

Consumers

While there is not much consumers can do to test the security of a mobile application, there are precautions that can be taken to safeguard personal data, to include:

- Choose your apps wisely. If an application wants access to information that it should not need, don't use that app.
- Be careful about permissions you grant. If a navigation application asks for access to your contacts, think twice about granting that request – especially if the app will not install without it.
- Update and periodically review your device's privacy settings.
- Always use unique passwords for different accounts. It is tempting (and easy) to use your social media accounts to sign up and sign in to a new app or site, but resist the urge.

The FCC has published "Ten Steps to Smartphone Security," a useful list of security tips for consumers. (https://www.fcc.gov/sites/default/files/smartphone_master_document.pdf)

HPE Security Fortify

HPE Security Fortify offers comprehensive SSA solutions featuring the broadest line of products and services in the market today. Fortify solutions help you build trust in the software you depend on by helping you find, fix, and fortify applications in less time and for less cost than comparable tool-centric strategies.

Learn more at:
hpe.com/software/Fortify

Sign up for updates

★ Rate this document

Hewlett Packard Enterprise

February 2016