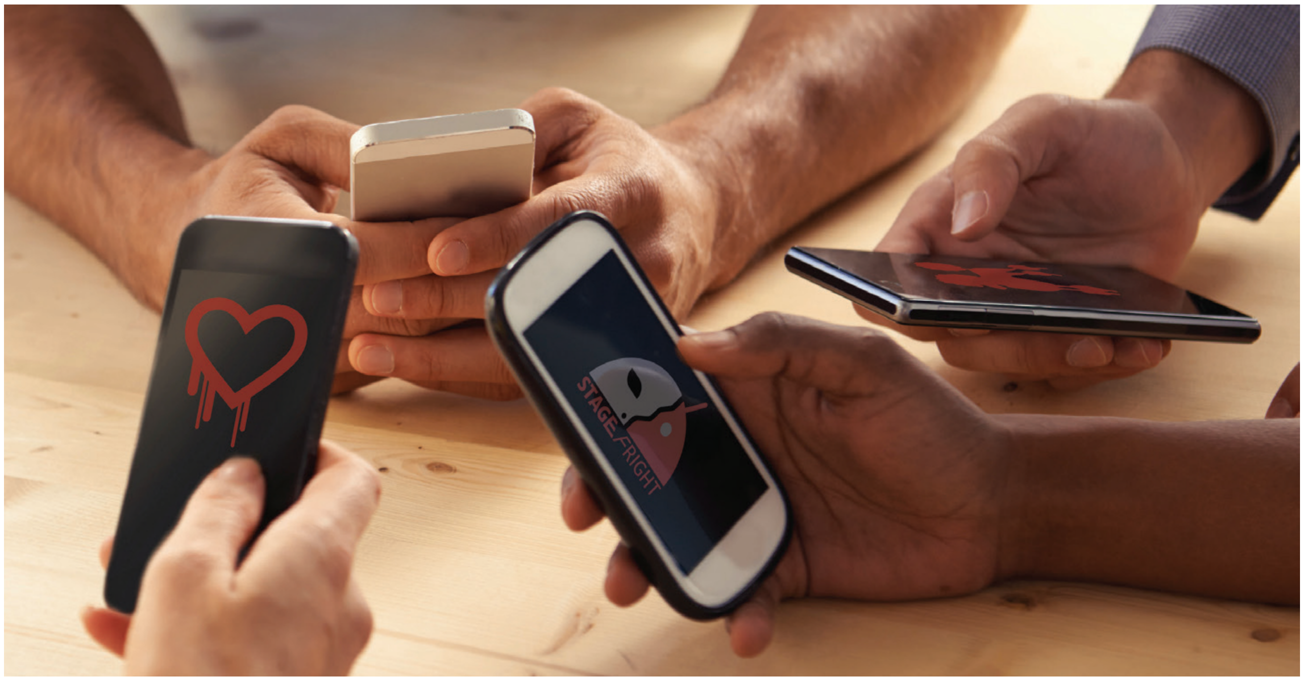
BLACKDUCK

OPEN SOURCE SECURITY & THE MOBILE ENTERPRISE



Employees no longer conduct business solely on corporate-issued computers and mobile devices. They routinely bring their own productivity tools through an organization's front doors, including smart phones, tablets, and notebooks. This shift in ownership from company- to employee-owned devices cuts costs, but also brings new security risks.

The use of open source software on employee-owned devices is ubiquitous, as it is across the enterprise. And companies have good reason to be concerned about open source security. A key example lies in the Android platform, which is built upon an open source code base. While Android boasts its own defenses, these defenses are easily bypassed with a few taps by users intent on getting their jobs done (as opposed to observing good security practices). Moreover, Android is subject to range of severe security threats, including the recently-disclosed and easily-exploitable Stagefright vulnerability.

Poor or incomplete corporate policies for mobile use, combined with vulnerabilities in the project that comprises Android (along with other open source software) can leave gaping holes in enterprise and mobile application security. This paper provides context for the current issues in mobile enterprise and application security and offers concrete tips for making organizations' mobile platforms more secure.

THE ENTERPRISE MOBILITY THREAT LANDSCAPE: AN OVERVIEW

Today, organizations face many challenges to securing the disparate mix of devices with access to corporate networks and data. Ever-increasing use of both stand-alone mobile apps and those that traverse the corporate firewall carries a growing risk of exploitation. These apps, like the enterprise infrastructure they touch, build on hundreds of open source components with a range of security profiles.

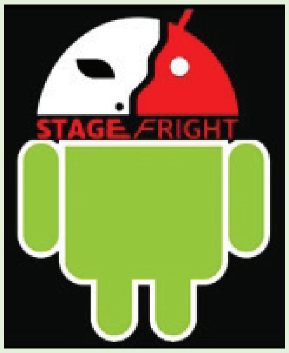
To understand the enterprise mobility threat landscape, let's consider the main areas of concern: mobile platforms, mobile apps, the data they access, and the APIs they use.

MOBILE PLATFORM SECURITY

The mobile platform landscape is chockablock with open source software. The most visible and pervasive open source mobile platform is Android, which is itself an aggregation of numerous open source components, starting with the Linux kernel, and including the Bionic C library, MySQLite, Binder, and the application, display, telephony, and multimedia frameworks that constitute the Android platform. Not only does the platform itself rely heavily on open source software, but so does much popular Android middleware, along with the development tools used to build both apps and the platform itself.

Unfortunately, while Android is the most popular and ubiquitous mobile platform with 78 percent market share as of 2015, according to IDC, it is also the smartphone and tablet operating system most subject to security risks – especially vulnerabilities in core components and malware masquerading as valid apps. On one hand, Android presents application developers with both a "sandbox" that separates applications from the platform, and "containers" that isolate applications and data from one another. On the other hand, Android also provides mechanisms to bypass the sandbox (e.g., native apps) and varying definitions of containers and the accompanying configurations and permissions that should govern application access to critical resources. The most egregious result of these weaknesses to date has been Stagefright, the open source native Android audio-visual (AV) player, in which a set of vulnerabilities exposes this central component to exploit via the AV media itself.

SHINING A SPOTLIGHT ON STAGEFRIGHT



Estimated to affect some 950 million Android-based phones and tablets, the Stagefright security flaws are among the worst Android vulnerabilities discovered to date.

Since Stagefright is a native Android application written in C++ (vs. the "Dalvik" Java dialect of most Android apps), it lacks the protection afforded by the normal runtime "sandbox" – running as a native Linux app vs. a Java application within a de-privileged virtual machine. These problems are exacerbated by Stagefright being granted "excessive privileges" for file access and execution beyond what is actually necessary to play audio-visual content.

The vulnerabilities in Stagefright date back to Android version 2.2 and persist up to more recent, widely-deployed versions. While patches are available, these vulnerabilities are likely to persist for some time, due to:

- Highly variable practices by both device manufacturers and operators in delivering updates over-the-air (OTA) to devices in the field
- Huge version proliferation of Android software components, and manufacturer and channel-specific fragmentation of the Android platform itself. Between device models and versions, there can be hundreds of various incompatible instances of Android deployed at any one time; moreover, many Android-based devices are never updated at all during their fielded lifetimes.

But Android is not the only mobile platform that integrates open source. iOS, while packaged as proprietary commercial software, builds on a range of open source projects, starting with an underlying BSD/Darwin UNIX kernel, and including application libraries (libc), utilities (e.g., bash) and many tools and software components needed to build applications.

Among other mobile platforms, there is a predominance of proprietary code with occasional outcroppings of open source, e.g., legacy SymbianOS (released as open source software and called OpenSymbian), portions of BlackBerry rebuilt around QNX (which runs many open source packages and has an open-source-like license), any phone running Java, and the recent release of key portions of Microsoft .NET.

Organizations rolling out an enterprise mobility program should at least be aware of the vulnerabilities that crop up in their mobile platforms and make a concerted effort to keep devices up to date, even if they cannot selectively patch and release mobile operating system versions. Larger enterprise IT shops, the integrators that serve them, and even small- to medium-sized organizations, however, are certainly in a position to benefit from intelligence on distinct software components (libraries, utilities, etc.) that directly impact mobile applications they develop, deploy and/or maintain themselves. But in today's mobile marketplace, users must go through device manufacturers and wireless operators to remediate threats via over-the-air (OTA) updates.

MOBILE APPLICATION SECURITY

With the introduction of Apple's iPhone App Store, the Google Play application marketplace, and dozens of other app stores and exchanges, the universe of mobile apps has expanded explosively to encompass several million distinct apps. This exciting but uncontrolled expansion has been accompanied by a growing constellation of security threats, including:

- Malicious apps snuck into poorly-curated app stores that attack other apps, exfiltrate user information and keystrokes, and participate in botnets and DDoS attacks
- Vulnerabilities in infrastructure, principally embedded browsers, libraries, and middleware (including Stagefright), used by marketplace apps and also by enterprise-developed applications
- Weaknesses in architectural elements intended to defend applications, including authentication, execution sandboxes, and containers

Consequently, enterprise IT organizations and other end users need to defend their employees and their networks against threats that accompany downloaded mobile software. They also need to be concerned about key aspects of applications they themselves develop and deploy, such as:

- Threats from malicious outside apps
- Exploits to client-side code that runs on mobile devices
- Vulnerabilities in host-side code in corporate data centers and the cloud, including both enterprise applications and mobile device management portals

Best practices for defending against downloaded apps include black-listing known bad software (and white-listing vetted apps), and running untrusted apps in special containers or even separate mobile virtual machines.

For in-house applications incorporating open source software, organizations are best served by implementing Open Source Hygiene – the cross-referencing integrated versions of open source components with databases of known vulnerabilities (NVDB, OSVDB, VulnDB, et al.).

DATA SECURITY: HOW IT RELATES TO THE MOBILE ENTERPRISE

Discussions of security practices tend to treat data as completely separate from the code that accesses and manipulates it. Best practices should include encrypting stored data and data streams, such that even if software components exhibit exploitable vulnerabilities and the enterprise perimeter is compromised, corporate data remains secure, right?

Well, not exactly. The context of the data in question ultimately determines whether it's subject to threat, and also whether vulnerabilities in code, including open source software, will impact the security of that data.

Any discussion of data security must consider three key contexts:

DATA-AT-REST: data stored on a rotating media or flash memory, in a phone or tablet or on a computer. Data-at-rest can be encrypted on a per-file basis or stored in an encrypted file system or volume. Naively, users often assume that encrypted data-at-rest is by definition secure, but it is actually subject to a range of threats:

- Data encrypted with insufficiently strong encryption or exfiltrated in a system breach can be decrypted remotely using brute-force decryption methods
- Data stored in temporary files, pre-encryption or prior to deletion, can exist as plaintext and is also vulnerable
- Weak authentication can lead to password theft and subsequent access to encrypted data-at-rest
- The varied software components that access and manage data at rest – such as encryption libraries, file systems, and applications themselves – are subject to a range of potential vulnerabilities, exposing the data they secure and manage to exploits

DATA-IN-MOTION: data traversing wireless and wired networks among mobile devices, data centers, and other Internet locales. As with data-at-rest, since encryption exists for data transmission, end-users and IT professionals often assume (erroneously) that data-in-motion is inherently secure. While wireless communications protocols support some level of encryption (e.g., WPA), the end-to-end transmission, even if otherwise encrypted, may not be secure, or not encrypted at all. Related issues include:

- Standard, presumed strong encryption methods have exhibited numerous, severe security vulnerabilities. In particular, several dozen exploitable vulnerabilities have surfaced in 2014 and 2015 in OpenSSL, the industry-standard open source implementation of SSL/TLS used to protect browser-server connections and other data exchanges. Most famously, these vulnerabilities included Heartbleed, Poodle, Freak, and CVE-2015-1793.
- Many vulnerabilities affecting data-in-motion surface not in the code itself but in the protocols they implement. One example is "Logjam," an exploitable vulnerability that causes negotiation of SSL communications sessions to fall back to 1990s-era weak encryption implemented as a concession to export restrictions then in effect.
- Unencrypted streams, or compromised encrypted ones (as above) are subject to a range of exploits, from wholesale interception, to packet sniffing to steal passwords and spoof authentication, and man-in-the-middle attacks.

DATA IN USE: plaintext (unencrypted) data residing in program variables, arrays, buffers, and CPU registers in the context of application and/or system software execution. Data-in-use is secured by the isolation structure offered by mobile platform, via Linux processes, Android containers and/or full-blown virtual machines. However, data-in-use is vulnerable when applications and system software are compromised through other means – root kits, boot code exploits, and a range of application-level exploits.

SECURITY OF APIS & MOBILE WEB INTERFACES: WHAT YOU NEED TO KNOW

While enterprise mobility security focuses on mobile devices themselves, it is equally imperative to secure the channel by which corporate and third-party apps communicate with back-end applications running in corporate data centers and/or in the cloud. Comparably, many mobile apps eschew specific interfaces in favor of SSL/SSH-encrypted web sessions running in browsers on mobile devices.

Designing, enabling, and supporting web APIs and/or libraries to enable VPN connections requires a rigorous command of what those interfaces expose on the devices and on the web. Equally challenging is securing browser-based applications from associated threats – Javascript malware, strong encryption by-passes (Heartbleed et al.), cross-site scripting forgeries, buffer overflow exploits, and myriad other risks.

As with other parts of the enterprise IT portfolio, the software that implements web APIs and web applications is built with open source, and/or heavily dependent upon it. Given the high-profile, mission-critical role of open source in enabling remote access, it is imperative that organizations fully catalog the open source components and versions in use on “both ends of the wire” and engage in Open Source Hygiene to discover vulnerabilities in those components and prioritize remediation.

SECURITY STARTS WITH VISIBILITY – KNOW YOUR CODE

Leveraging employee-owned devices to conduct company business starts out as a win-win exercise, but without good policy and practices, can result in escalating security threats. Key is choosing solid, well-maintained, and secure open source components early in the mobile application lifecycle, and staying ahead of emerging threats throughout development and deployment.

Black Duck Software provides the tools and capabilities to help organizations create and deploy secure applications using open source, and to mitigate emerging risks from vulnerabilities in code that originates with open source communities. The Black Duck Hub and the Black Duck® Suite help security and development teams uncover and mitigate open source related risks across application portfolio by:

- Scanning and identifying open source software throughout organization code bases
- Mapping known vulnerabilities in open source organization code integrated and deployed in enterprise applications
- Consolidating vulnerability information to aid in prioritizing, scheduling, and tracking remediation
- Monitoring for newly disclosed vulnerabilities in open source code

Want to learn more? Start securing your use of open source software today with our free 14-day trial of the Black Duck Hub.

BLACKDUCK