THE GREATEST RISKS IN MOBILE APPS

By Anthony Bettini

Due to the lack of advanced protective mechanisms such as sandboxing and broad use of code signature enforcement by the Microsoft Windows operating system, vulnerabilities accidentally inserted into code by programmers have traditionally presented one of the greatest information security-related risks within the enterprise. In contrast, the greatest risks associated with mobile apps are not related to insecure programming mistakes, but rather to the intentional misuse of supported features, which put the enterprise and sensitive data at risk.

Software is an enabler. Software has allowed new businesses to be successful, governments to run more efficiently, and people to improve their lives. But software has also introduced new risks to all of these parties and counter-parties. The ability to measure risk from software has evolved significantly in the past decade. Ten years ago, when looking at software vulnerabilities or "threats," people in the commercial sector would primarily focus on the absolute impact from the worst possible outcome associated with exploiting the vulnerability.

While focusing on absolute impact is still common, times have changed, and more researchers are regularly focusing on attacks first. By better understanding attacks, we can understand where our resources are best focused to mitigate the risk of an attack succeeding and to mitigate the damage done when an attack does succeed. As all systems of importance will be attacked, the architecture of the system appears to have a strong causal effect on the attacks that occur against that system.

Mobile is Different

To understand why mobile is different from the security perspective, it is helpful to refresh our understanding of security for the Microsoft Windows platform. The vast majority of commercial sector security solutions have been designed around improving the security for Microsoft Windows deployments. Whether the product has been a "next-generation enterprise firewall" or an anti-virus engine, it was likely built with the goal of better protecting Microsoft Windows desktops and servers. Security products have been built this way because within the enterprise, desktop and server deployments have traditionally been almost entirely Windows-based. Use

of Windows in the consumer space has been similarly popular, with more than 90 percent of consumers using Windows as their desktop OS. Prevalence has therefore dictated the focus, both for attackers as well as for those charged with defending systems and networks.

OS Architecture

Fast-forward to the present, where mobile is being adopted at a fast pace and mobile devices are running operating systems with drastically different architectures from Windows, and we can begin to see why the risks are different and why different solutions are needed. Traditionally on Microsoft Windows systems, sandboxing has not been used, code-signing requirements have not been broadly adopted, and the OS has generally been left "open." This has allowed malware to spread freely, causing the growth of the modern anti-virus industry.

Though the Android OS is similarly open and plagued by malware, Apple iOS implements a number of controls designed to prevent the execution of malicious or otherwise unapproved code on devices. These controls include static and dynamic code signature enforcement, which ensure that only code signed by Apple can be executed, as well as a process sandboxing model that allows for fine-grained control over which files and other resources a process can have access to. Additionally, iOS disallows execution on the stack, and supports Address Space Layout Randomization (ASLR) as well as Position Independent Executable (PIE)-compiled executables, which when combined make successful exploitation of vulnerabilities very difficult to accomplish.

Apple also requires that all apps sold via its app store be reviewed prior to distribution. This review process is not widely documented but does include at least checks related to trademark violations, copyright infringement, and the use of undocumented APIs, as well as some basic security-related checking. The OS security features combined with this review process have thus far been generally effective at keeping malware off of the iOS platform. With all of this said, the enterprise still faces significant risk from mobile apps.

Software Procurement

Although not exclusive to mobile, another large change that is most clearly seen in the mobile space is the complete change in software procurement (from the perspective of the enterprise). Traditionally, enterprises would procure software from large vendors, which came with some inherent level of trust. Users within the enterprise were often restricted from installing software that didn't come via the IT procurement team. With the "Bring Your Own Device" (BYOD) movement and the consumerization of IT, this software procurement process for mobile has completely flipped. Now, enterprises in the commercial sector are embracing BYOD, finding it's actually cheaper for the enterprise if users buy their device because users take better care of it and fewer support calls reach IT.

But as users bring their own devices, they are also bringing their own software. Instead of the software being written by a vendor and going through an IT approval process, mobile software is often being written by small development teams (including many one- or two-man shops) in jurisdictions outside the United States.

Intentional Behaviors

OS architectures greatly influence the attacks against the system and the attacks greatly influence the necessary defenses. On Microsoft Windows systems, we primarily see attacks involving either A) malware or B) unpatched vulnerabilities. In the case of the unpatched vulnerabilities, these are ultimately insecure programming mistakes (buffer overruns, heap corruption bugs, integer wrapping mistakes, etc.). Thus, to protect Microsoft Windows deployments, a large amount of resources needs to be spent on vulnerability mitigation via detection, patching, and API hooking in-memory. These programming errors are unintentional.

In the case of mobile apps today, the sources of risks in the software are actually intentional behaviors. That is to say, the developers added a feature or piece of functionality into the mobile app on purpose, which put the enterprise or consumer at risk.

Examples of Intentional Behaviors in Mobile Apps

At Appthority, we've examined hundreds of thousands of mobile apps for behaviors that could place an enterprise at risk. The following are a small handful of the risky intentional behaviors we've found in mobile apps:

In the case of mobile apps today, the sources of risks in the software are actually intentional behaviors.

Malware
The placement of malware in a mobile app is intentional. This isn't much different than malware on the desktop, other than to say that malware evolutions and trends seem to be happening far faster on mobile than they did on the desktop.

Transmitting Contacts
Developers, especially developers of social networking apps, often transmit the contacts or address book from the device. The reason, while likely varying, generally relates to increasing the viral or network effects of the app. In other words, the developer wants to use the owner's contacts to expand his or her customer base. There are a lot of problems with this approach. For one, whose contacts are they? For instance, if a consumer buys an iPhone and plugs it into his or her corporate desktop at work, it will likely sync with the contacts from Outlook. Those Outlook contacts are likely enterprise contacts, which is corporate data owned by the enterprise. Ignoring the fact that most apps that access the contacts on mobile devices don't even ask for permission, if the app did ask for permission, it would be the user saying "yes," even though the contacts would be enterprise data.

Due to the recent public outcry over an app called "Path" taking users' contacts without permission, developers are now starting to look into hashing the contacts and then transmitting only the hashes. This is an improvement, but certainly only a start. It is important to note that transmission of the contacts is only an example; the larger issue is enterprise data sitting on a consumer-managed device with apps that have behaviors that affect the security of that data.

Location Tracking (With or Without Permission)
Everyone generally understands what location tracking is and at least some of the risks associated with carrying around a computer in your pocket that knows your location at all times. However, we're finding that location tracking is being done a lot of different ways, by a lot of different entities. For instance, maybe an enterprise would agree to a device being location-tracked by a carrier, but a user may not make the same choice.

Location tracking also is happening in a wide variety of ways, such as via cell tower triangulation, GPS, GeoIP, etc. Some of these types of location tracking, on certain mobile platforms, require explicit permission. But does the user have the right to grant the permission if the enterprise purchases the device? We're also seeing apps that intentionally bypass the permission model to track users without their permission. For instance, on Android, there exists a permission to track location by GeoIP. However, that permission is only required if the app leverages the API provided by Google to track location by GeoIP. We've seen apps that implement their own HTTP clients to leverage third-party GeoIP providers exclusively so the app developer can location track users without their permission.

Below we can see the request generated by a mobile app in order to obtain the Internet-facing IP address of the device being used. This information could be used in a way that violates corporate policy, and because of this, this app's behavior should be subject to review prior to use in the enterprise.

```
GET /my-ip-address.php HTTP/1.1
Accept-Encoding: gzip
Host: www.ipaddresslocation.org
Connection: Keep-Alive
User-Agent: Apache-HttpClient/
UNAVAILABLE (java 1.4)
```

Conclusion

The trend to move to mobile will not be reversed. Mobile, Cloud, and big data truly are changing modern computing for both enterprises and consumers. While change brings risk, modern technology allows enterprises to begin to measure that risk. Any software running on a mobile device, whether from a public app store or a private enterprise app store, needs to be inspected and validated against corporate policy, as that software can put the enterprise at risk. Even in BYOD environments, software running inside the enterprise should not be a black box. Q

Anthony Bettini *is the CEO at Appthority, the leader in mobile app risk management solutions and recent winner of the most innovative company of the year award at RSA. Anthony's current work focuses on scalable software analysis and behavioral profiling of mobile apps. His security experience comes from working for companies like Intel, McAfee, Foundstone, Bindview, and Netect. His background is in vulnerability research and he has published new vulnerabilities in Microsoft Windows, ISS Scanner, PGP, Symantec ESM, and other popular applications.*