

Securing Mobile Devices: Present and Future

McAfee® Labs™ examines the current state of smartphones and other mobile devices and the security risks associated with their new capabilities. We also predict near-term future threats for these handy machines.

By Dr. Igor Muttik

Despite steady progress in securing desktop computers—using safer hardware, operating systems, and applications—malware is not going extinct. With today’s explosive proliferation of smartphones, tablet computers, and other mobile devices, we have to wonder whether our pocket devices can also be secured. We might assume from our extensive knowledge in protecting desktop computers that the new wave of mobile hardware should be relatively secure because we shall benefit from the lessons we have already learned. In this paper we shall examine and describe in detail why this is unlikely to be the case.

We won’t keep you in suspense: Mobile devices are not going to be less susceptible to security problems. The overall threat of malware might decline, but the damage to mobile devices is likely to be high because smartphones are always connected, they always carry some personal data, and they are even equipped with small cameras, microphones, and positioning devices—just like the spies carried in old movies. The wider choice of built-in devices compared with desktop computers (or laptops and notebooks) makes the operating systems (OS’s) and applications more complex and ultimately increases the attack opportunities.

The core of contemporary OS’s, such as iOS and Android, is based on Unix/Linux, which makes the system reasonably secure. However, due to the competitive pressure of the market, the manufacturers frequently prioritize time-to-market concerns over security and may add core drivers and applications that are insufficiently tested. There could also be flaws in implementing updates for the firmware and OS that would make the robustness of the OS core irrelevant—as an attack may be able to entirely replace it.

Nearly all the types of threats to desktop computers that we have seen in recent years are also possible on mobile devices. (Parasitic viruses may be a notable exception for modern mobile OS’s, more on this below.) Moreover, we are bound to see threats readapted to mobile environments and, unfortunately, we are also likely to see new kinds of malware that target smartphone capabilities that are not available on desktops.

In this paper we shall use mobile devices and smartphones interchangeably. The first term has a wider scope (as it includes pocket game consoles, readers, tablets, etc.) but the overlap of features is great and contemporary phones offer pretty much everything that other more specialized devices do.

Table of Contents

Specifics of mobile devices	4
Environments of the past	7
Android	8
iOS, Windows Phone 7, and others	12
Current and emerging ecosystems	12
Changes and predictions	16
Conclusion	26
Acknowledgements	27

Specifics of mobile devices

Mobility brings vulnerability

Mobile devices are on the move, meaning they can more easily be lost or stolen and their screens and keyboards are easier targets for “over the shoulder” browsing. Even if a device is properly secured with a PIN or passphrase, a determined attacker has a reasonable chance to intercept the PIN when the owner enters it. The rest of the compromise depends on the pickpocketing skills of the thief or the carelessness of the owner. Even a short lapse of control over a phone may allow a compromise.¹

Most of us prefer to have a single smartphone with multiple functions rather than carry around several devices. Of course, that makes these devices more versatile, expensive and, consequently, more attractive for thieves. They may be interested purely in the hardware but, increasingly, they also want access to the device’s data, which can be very valuable (and may be worth a lot more than the hardware).

Contemporary security measures make it harder to resell stolen devices. (In many countries reporting a lost phone’s IMEI (International Mobile Equipment Identity, a unique identifier) would allow the supplier to block further use of a device; unfortunately this is not the case in some countries.) Reprogramming the IMEI is possible, but it may be illegal in several countries to possess such equipment. Although the cost of devices goes down each year, the cost of the data they carry may do just the opposite. Today in many cases the data extracted from a phone is more valuable than the cost of the parts. This value will only grow as smartphones are increasingly used for banking (including near-field communications) and business (storing trade secrets, blueprints, roadmaps, patent documentation, etc.).

Battery powered

Mobile devices require periodic charging. More intensive computing tasks and communications (for example, using wireless networking or making calls) result in an accelerated drain of the battery. To conserve power, phones enter an energy-conserving “sleep” mode. In this mode the software uses a timer to wake up periodically and do necessary tasks. But some applications may prevent a mobile device from entering sleep mode. Poorly written software can also accelerate battery discharge. Naturally, when malware operates it will also tax a battery—we saw this in field cases of the SymbOS/Cabir and SymbOS/Commwarrior worms, which actively used the Bluetooth interface for propagation. Malware writers generally care much less about the user’s experience than legitimate software manufacturers, so malware may wake up often and drain the power at an accelerated rate. One exception can be malware that conceals its presence for a long time, as, for example, advanced persistent threats (APTs) do. In this case malware authors may take measures to conserve power to avoid attracting a user’s attention. Restricting the demand for energy-costly operations may prolong malware’s lifetime, which sometimes could be the attacker’s goal.

At BlackHat 2011 we saw a researcher compromise battery firmware—something that definitely, as a minimum, could cause a denial-of-service (DoS) attack when a battery is “bricked” (made useless).² As an example, a piece of malware could target a person (or a group) responsible for handling some emergency situation: If malware such as a remotely controlled botnet or an APT designed for warfare could brick a battery following a remote command, the affected person or team would be left without a convenient method of communication (and possibly the only one available).

GPS

Tracking the owner

Many mobile devices employ global positioning systems (GPS) to provide applications with the device's current position and allow location-specific services (for example, showing a local map or marking photographs with the place they were taken). This means, of course, that the software knows the location of the device and usually the device's owner. It is easy to imagine how exposing or leaking this information could be a problem. We have already seen in the wild the tracking malware GPS Spy (or TapSnake) for Android. There must be efficient controls ensuring the privacy of phone owners and limiting the distribution of GPS data. (Contemporary OS's for mobile devices, such as iOS and Android, provide certain mechanisms limiting access to GPS data.)

The careless handling of GPS data by the application or the OS could easily create a privacy problem. This was spectacularly demonstrated by an incident in which iPhones' GPS locations were synced to a desktop and stored unprotected in the file consolidated.db (which was updated during every sync operation and could be seen by anyone with the access to the file). This was a serious privacy loophole that Apple fixed fairly quickly.

Tyler Shields of Veracode demonstrated a spying app for BlackBerries that can track the location of a phone.³ The use of this type of malware could assist, for example, in kidnapping high-profile targets.

If users choose to automatically share GPS data (for example, by using a service such as Foursquare, a location-based social networking system), then they willingly give up their privacy.⁴ If, however, the provider is compromised and leaks tracking data for many users, then the privacy loss is multiplied.

Mobile devices carry location data even when GPS hardware is not in use. The network knows which specific cell (mobile tower) any device connects to. This information is less precise than GPS coordinates but still generally pinpoints a phone with reasonable accuracy (perhaps within 100 meters). However, this data may be available only to the OS; most applications should normally not have access to this data (unless they manage to elevate their privilege by employing some vulnerability in the OS). So, there may be OS controls to disable GPS hardware, but they are not guaranteed to be 100 percent effective due to this network data.

Tracking thieves

In theory, a GPS in the phone could allow the owner or law enforcement to track stolen devices. The problem is, of course, that to track thieves we must be able to record GPS data remotely. If such mechanisms existed in the phones, we would be able to locate thieves; yet that ability to track would be in conflict with owner's privacy. Solving this problem requires the reliable identification of the device's owner. Most common methods (PIN or passphrase) are not as good as, for example, biometric authentication, which is likely to stay prohibitively expensive for some years.

Camera and microphone

Smartphones can capture photographs, video, and audio. This functionality is also available in portable game consoles and tablets. These devices are software controlled, and in many cases it is possible to operate them even when the user is not aware that the camera or the microphone is active.

Most OS's offer controls to disable the camera and microphone, but successful exploitation of vulnerabilities in the OS (resulting in the elevation of privilege to control the hardware) would allow malware to snoop on users and their surroundings. Such vulnerabilities are regularly discovered and exploited.⁵ We've already seen Trojans that record phone conversations and send the recording to another location.⁶ Law enforcement could also try to track smartphone thieves using sounds and images from a phone.

3. <http://www.veracode.com/blog/2010/02/is-your-blackberry-app-spying-on-you>

4. [http://en.wikipedia.org/wiki/Foursquare_\(website\)](http://en.wikipedia.org/wiki/Foursquare_(website))

5. <http://www.cs.ncsu.edu/faculty/jiang/GingerMaster/>

6. <http://blogs.mcafee.com/mcafee-labs/latest-android-malware-records-conversations>

Implications of multi-CPU (or multicore) design

Smartphones and other portable devices use their computing power to serve the user (screen, keyboard, applications) and to maintain baseband communications (mobile signal: GSM, 3G). The last operation requires processing in real-time. (If received or transmitted data is not processed quickly enough, the phone will suffer communication interruptions, resulting in a very poor user experience.) To keep up, devices may use two processors (or they are implemented as a System-on-Chip bundle, which may include multiple computing subsystems in one SoC chip). We anticipate in the future that manufacturers may further split the functionality, for instance, adding SoC cores for graphics and security.

As a result of these design decisions, the CPU subsystems may become isolated, which will pose some problems in maintaining safety. Ralf-Philipp Weinmann has already demonstrated successful attacks on baseband CPUs (Qualcomm CPU in HTC and Infineon in iPhone).⁷ Another example shows the compromise of a communications subsystem (via malformed Bluetooth) may be completely invisible to security software running on a separate CPU (or on a separate or dedicated core).⁸ In other words, separate CPUs may be exploited separately, and even detecting such situations may be difficult.

Other mobile devices

Tablets

Today's tablets are more powerful than laptops were a few years ago. Although their lack of real keyboards makes them unsuitable for many tasks (editing texts, programming, and design), they are very suitable for browsing the web, which today is the primary source of malware. This malware frequently enters computers via vulnerabilities in browsers (and their plug-ins, such as Acrobat, Flash, and Java) so we must expect that trend to continue on tablet computers.

We needn't view tablet computers separately from mobile phones. Tablets mainly differ in the size of the screen, but they share the same software, OS's, and CPUs so their security concerns are nearly identical. About the only difference is that some tablets (for example, Toshiba's Thrive) can operate as a USB master, which increases the attack surface of such devices.

Smart media players (MP3 and video)

Music players are not normally equipped with browsers; that removes a lot of exposure to malware. The dangers start if these devices have networking capabilities and are programmable. In this environment malware can exist.

However, music players are not likely to attract attackers because they do not normally carry sensitive data. Using malware to steal music or videos is not yet popular, as these materials can be fairly easily pirated. However, with the proliferation of digital rights management (DRM) restrictions the situation might change. Attacks targeting entertainment (music, films, games, etc.) may become more economically viable. We would expect these attacks to target devices that do not support DRM but still carry some valuable media. Other possibilities include attacking the DRM itself via weaknesses in its implementation or hardware, and man-in-the-middle (MITM) attacks during key distribution.

Another malicious use for such devices could be to establish an entry point into a computer or a network. It may be easier to compromise a computer via a connected MP3 player than by doing the same remotely via a network. We could see an attack similar to W32/Stuxnet, which used a LNK vulnerability to auto-execute from USB sticks. A connected media player would frequently use a richer interface than a USB stick, thus providing a larger attack surface.

Attackers could also abuse a media device to establish malware persistency or propagation. Malware worms may spread by copying themselves onto all available storage devices and spread (or reinfect a cleaned computer) via media devices as they would via USB sticks. We know of well documented cases of distributing malware via infected USB sticks. We can also imagine media players or tablets used for that same purpose. If one finds a shiny media player, it might be hard to resist the urge to connect it. Such devices can be used for targeted attacks too.

7. <http://blogs.mcafee.com/enterprise/mobile/27th-chaos-communications-congress-mobile-security-and-more>

8. <http://www.technologyreview.com/computing/35094/>

Portable game devices

Game consoles are basically computers. Many now are networked and provide Internet access. One feature that makes portable gaming devices (such as Nintendo DS, Sony PSP, etc.) potentially more dangerous than other mobile devices are peer-to-peer (P2P) functions (direct device-to-device communications, usually to support multiplayer games). If these connections allow the transfer of programs (which is how the Cabir worm spread on the Symbian platform), worms would be easy to implement. Even if P2P communication is limited to data transfers, exploiting vulnerabilities may allow remote code execution and viral propagation.

New game devices also frequently use “nongaming” technologies. One example is PlayStation Vita, which incorporates 3G, Wi-Fi, Bluetooth, GPS, cameras, and microphones.⁹ The gulf between gaming devices and mobile devices is rapidly shrinking.

Diversity of the devices

One barrier to malware proliferation is the diversity of mobile devices. Because there are many hardware models and OS's, targeting more than one device becomes more costly. We expect, however, to see increased consolidation and standardization in both hardware and software. Early news about the upcoming Android Ice Cream Sandwich (the next Android OS to cover both phones and mobiles) and Windows 8 (to work with multitouch screens and ARM CPUs) fully supports this.

With standardization comes the danger of a monoculture (think Windows on the desktop), which is more susceptible to wide attacks simultaneously affecting multiple targets.

Diversity, on the other hand, prevents large-scale attacks but is still suitable for targeted malware. Attackers can compensate for the limited number of targets by increasing their potential returns. These can be boosted by directing attacks at highly valuable targets or by increasing the time malware stays undetected. Stealth (or rootkit) capabilities as well as complicated detection and removal could play an important role in achieving those goals.

Environments of the past

From the beginning, mobile devices became targets of malware. Proof-of-concept attacks on the Symbian OS eventually led to field infections and the further escalation of malware risks.

Symbian malware

Smartphones were first developed by Nokia and Ericsson in 1996-1997. They were marketed as “smartphones” around 2000 and relatively quickly standardized on the Symbian OS. Most commercially available phones ran it. At its peak, Symbian was installed on more than 350 million devices.

The first malware proofs of concept for mobile phones took advantage of insecurities in Symbian OS. (And before Symbian there was a handful of malware for the Palm OS.)¹⁰

Software for Symbian phones comes in a software installation script (SIS) package. There are many ways to distribute them—via Internet downloads, email, SMS, Bluetooth, infrared, SD cards, PC connection, and over the air.¹¹

The first in-the-wild mobile virus was reported in 2004: It was a Symbian worm called Cabir and it propagated via Bluetooth.¹² Cabir was followed by lots of other malware families; most were Trojan horses and we have seen hundreds of them since 2004. The majority of this malware was distributed over the Internet.

9. http://en.wikipedia.org/wiki/PlayStation_Vita

10. <http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=98801>

11. http://www.developer.nokia.com/Community/Wiki/Deploy_SIS_file_on_hardware

12. <http://www.f-secure.com/weblog/archives/archive-012006.html>

J2ME malware

J2ME, now known as Java ME (or JME), is a Java-based programming environment designed to operate on mobile devices and embedded systems.¹³ At one time there were more than 2 billion such devices, but today J2ME is being replaced by more modern OS's such as Android. (The core of Android, just like that of J2ME, is a virtual machine based on Java.)

The wide distribution of J2ME from 2006 to 2010 attracted a corresponding surge in malware for this platform. Particularly popular were Trojans that made calls on premium lines and sent expensive SMS's.¹⁴ At some point there was so much J2ME malware that the total count exceeded that of Symbian malware.¹⁵ (Many Java malware are language-specific Trojans. This type of malware was particularly common in Russia due to the country's weak regulation of the telecommunications industry.)

The first widely publicized Android malware (FakePlayer.A) was basically the Android equivalent of a J2ME premium-rate SMS-sending Trojan. It is likely that the author modified existing code for Android and recompiled.

Android

Let's talk for a bit about Android. It is the fastest growing OS for mobile devices and it is already attracting a relatively high number of malware.

One of a kind

Android has a unique position in the market. It is free and, therefore, provides exceptionally attractive value for money! Vendors and mobile operators have much higher operating margins when selling and supporting devices based on Android. The OS is also open source, so it is easy to customize for new devices. It is no surprise that Android has taken the market by storm.

Android is the quickest-growing OS for smartphones. It was first released in 2008 and is already the market leader. Figure 1 shows OS market share at the end of 2010.

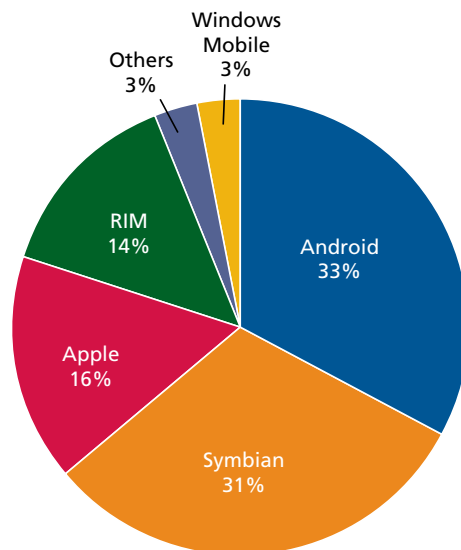


Figure 1: The Android OS has the biggest piece of the smartphone pie after just four years in business.

13. http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

14. http://www.securelist.com/en/analysis/204792080/Mobile_Malware_Evolution_An_Overview_Part_3

15. http://www.securelist.com/en/analysis/204792080/Mobile_Malware_Evolution_An_Overview_Part_3

Android's competitors, of course, are not likely to give up. One tactic to limit Android's success is by creating legal obstacles.¹⁶

Android not only leads in market share for the number of mobile devices; unfortunately, it also surpassed Symbian in the amount of new malware in the most recent quarter.

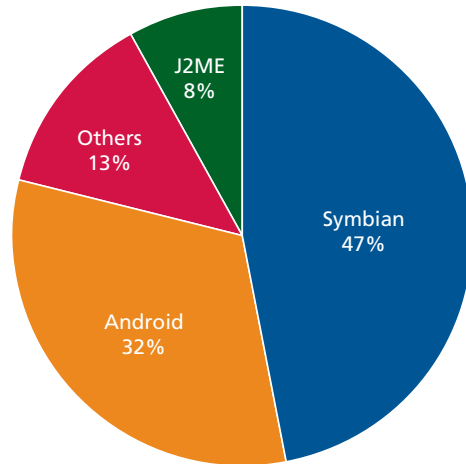


Figure 2: Historically Symbian has been the most popular mobile OS for malware writers. But Android is catching up quickly.

Figure 2 shows Symbian's overall share of malware (cumulative for all years) to be superior to that of Android's. If we look at only one recent quarter, however, then another aspect of Android's "market leadership" is apparent.

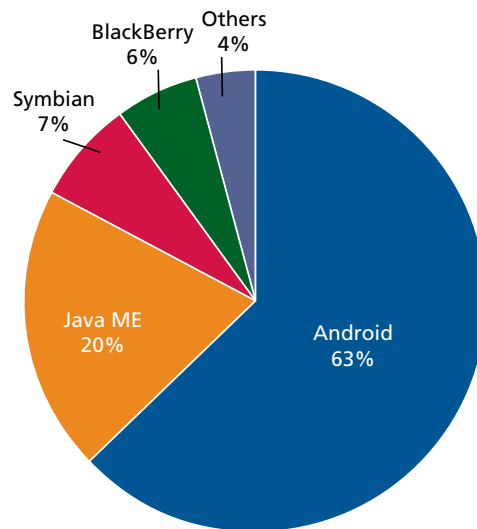


Figure 3: In the second quarter of 2011, the volume of Android malware was greater than that of any other mobile OS.¹⁷

We firmly believe the reason for this explosion in malware is due to a change in distribution methods. In the past, worms mostly hopped from one device to another (such as the Cabir worm propagated via Bluetooth); this method requires a large number of users in close proximity and their willingness to "accept" a transmission. Or, malware had to be downloaded from a Nokia site (or from some other site distributing Symbian programs).

Today nearly all infections come from application markets/stores (the global Android marketplace and popular third-party stores, some popular only in certain countries). There are many examples of recent Android malware. You can examine their descriptions at the McAfee Virus Information Library.¹⁸

Security model

The Android security model is simple: applications declare at installation the list of actions they intend to take (requesting “permissions”).¹⁹ The user has the choice to approve or reject this request (and only in its entirety, users cannot deny any selected permissions). Approving will enforce the declared limitations after the installation. Rejecting will block the installation. It is also not possible to control (neither deny nor grant) permissions after the installation or at runtime.

This model suffers for three reasons:

- It relies on the user’s making the right choice (and the description of many permissions looks cryptic even to technically savvy users)
- Programs sometimes demand more rights than are necessary, so users are accustomed to anomalous and “greedy” requests
- Users are likely to accept any permissions if they really wish to run the program, and their eagerness can be exploited by social-engineering methods

Security APIs

Android provides a small set of APIs to administer the device; the OS controls the password/PIN policies and can remotely wipe the phone.²⁰ Unfortunately, this set is fairly limited and is of little help when building a security product.

It would be highly beneficial for Android to support a standard set of general-purpose security APIs in the OS core (allowing access to kernel memory, and smooth interception of the file I/O and network data flows). Apps with such deep access would need to be signed in a special way. Another useful measure would be to build cryptographic authentication into the OS APIs so that security applications could trust the data they receive from the OS (and know it has not been tampered with). This step should reduce the rootkit problem on Android devices and thwart some MITM attacks.

Weaknesses

One DefCon 2011 presentation reported multiple deficiencies in Android design and documentation:²¹

- Conflicting and missing API descriptions affect the security of the applications due to developers making incorrect choices. In Android Version 2.2 only 78 APIs out of 1,207 were documented, and six of them were incorrect.
- Interprocess messaging in Android is either explicit (in which the name of the recipient is specified) or implicit (a custom message with no specific target, such as “my.special.action”). This distinction allows some messages to be intercepted or spoofed by malware, typically allowing data leakage or DoS attacks.
- Storage (SD cards) is world readable, and files (and folder names) will persist even after the apps that created them have been uninstalled or the device has been fully wiped
- Overprivileged apps (that request more permissions than necessary) are common (31 percent according to the authors). They violate the principle of least privilege. There are several causes of overprivileged apps:
 - » Poor documentation on which APIs need which permissions
 - » Testing artifacts: debugging code not removed after development or quality testing
 - » Errors propagated via forum advice: Developers keep copying and pasting the same substandard source code, even if it contains mistakes that the authors have subsequently corrected.

18. <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=501748>, <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=509500>,
<http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=522281>, <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=518925>,
<http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=501599>, <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=501639>
<http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=527859>

19. <http://developer.android.com/guide/topics/security/security.html>

20. <http://developer.android.com/guide/topics/admin/device-admin.html>

21. Y.Tsepenyuk O’Neil and E.Chin, “Seven Ways to Hang Yourself with Google Android.” DefCon 19, Las Vegas 2011

According to a report at BlackHat 2011 that was compiled from behaviors of live Android applications executed in a sandbox, 8.4 percent of apps send out IMEI data in unencrypted form.²² (IMEI is usually considered to be personally identifiable information, as most mobile devices are personal.) The same report showed that about 22 percent of apps requested access to the IMEI at installation. The difference between 22 percent and 8.4 percent reinforces the earlier assertion of “greedy” permissions.

Dalvik virtual machine

Android software comes in an APK package, which has a manifest describing desired permissions. Unless an Android application explicitly requests access to certain types of data or operations during the installation, it will not receive access. Once an app is installed, it operates in DEX format, which is executed in a Dalvik just-in-time Java-based virtual machine (VM).

The process isolation is provided via the OS because the Dalvik VM allows the execution of native code built in Android’s NDK; thus, it is not possible to achieve isolation through the VM alone.²³ All types of Android applications (pure Dalvik apps, native, or mixed) have the same security limitations.

According to the Microsoft Security Report Volume 11, exploits in Java were the most prevalent type of attack on Windows from the third quarter of 2010 through the second quarter of 2011.²⁴ Clearly, sandboxing based on Java is inherently insecure. Further, we believe that some of the exploit techniques might be applicable for attacking Dalvik in future. (The same report contains the curious observation that an Android exploit called Lotoor is frequently observed on PCs. It is detected on PCs while users download infected apps to transfer them onto their mobile devices, according to Microsoft stats.)

Application signing

Android applications must be signed, but there is no obligation for these signatures to be certified by any authority—so many apps carry self-signed certificates. This mechanism helps in tracking good software from the same source and builds trust in those sources. But it is not effective to track malware, as signatures for such applications can be generated for free, and new signatures provide no information about trustworthiness.

Just as with trusted digital signatures under Windows (called Authenticode), certificates of trusted sources for Android apps (even self-signed) do have a certain value. Thus we must assume that they will be targeted by malware authors. Stealing them will allow attackers to put the name of a trusted source onto their creations, which should ensure quicker malware distribution. If many digital keys are compromised, then the most popular might be used by the bad guys in targeted attacks to achieve a higher probability of success.

A group of security software manufacturers joined forces under the umbrella of the IEEE to implement the “IEEE software taggant system.”²⁵ (Taggant is a chemical added, for example, to plastic explosives. It helps determine which factory produced it and aids the analysis of explosive particles.) The IEEE taggant system automatically puts a cryptographically strong marker into software at the point of creation and allows the tracking of any program to its source. The advantage of the taggant system is faster scanning speed by security software and, from the developers’ point of view, taggants are free (unlike, for example, Authenticode signatures). The taggant system should be able to assist in telling bad software sources from reliable ones, but the system needs to be integrated into developers’ tools. The taggant system would not add value to iOS and/or Android as they already require app signing. But any other OS can take advantage of the system to integrate a secure reputation mechanism into its environment.

22. Neil Daswani, “Mobile Malware Madness, and How to Cap the Mad Hatters”

23. <http://developer.android.com/sdk/ndk>

24. <http://www.microsoft.com/security/sir/default.aspx>

25. http://standards.ieee.org/news/2011/icsg_software.html,

https://media.blackhat.com/bh-us-11/Kennedy/BH_US_11_KennedyMuttik_IEEE_Slides.pdf

iOS, Windows Phone 7, and others

Apple's iOS is currently the biggest rival of Android. Apple so far has done an excellent job of securing its devices; as we write this there were no reported cases of malware for iPhones that have not been jailbroken. (Jailbreaking opens the iPhone to unauthorized apps, posing a security risk because it allows any unsigned software to run.) We shall touch upon security risks in the iPhone world in following chapters.

Apart from Android and iOS, other operating systems may also play significant role in the future. As these OS's are under active development, we shall withhold our analysis of their security features until later. Here are the most prominent efforts:

- Windows Phone 7.²⁶ Nokia has decided to use this OS in their future devices. An IDC study predicts Windows Phone will reach about 20 percent of the market and shadow both Apple's and RIM's OS's.
- Samsung's Bada²⁷ has tiny market share, but Samsung partnered with Intel to build Tizen,²⁸ which is likely to subsume MeeGo²⁹
- WebOS³⁰
- Cloud-based OS's such as Google Chrome OS³¹

Current and emerging ecosystems

Let's discuss the environments in which mobile devices operate. The security of each device is determined not just by the robustness of its software but also by the entire ecosystem in which it works. From the security point of view, the ecosystem includes safeguards in hardware, the OS (and any other privileged app running as root), the OS update process, and external resources (for example, application stores) that a device contacts.

The security of mobile ecosystems

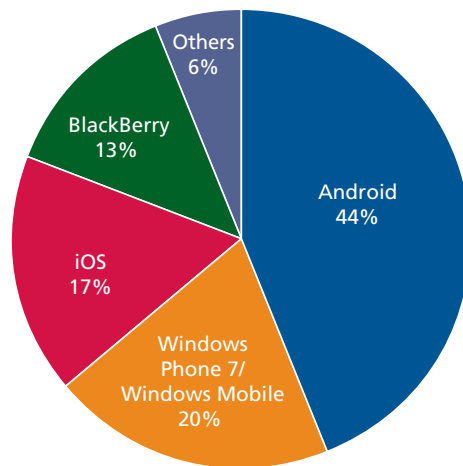


Figure 4: This IDC study from June 2011 predicts mobile OS shipments in 2015.³²

To properly compare the security of today's smartphones we cannot simply look at their OS's and evaluate the implementation of their security features.³³ Mobile devices are designed to connect; they operate within multiple networks (each adding to the risk) and download programs from the Internet.

26. www.microsoft.com/windowsphone

27. [http://en.wikipedia.org/wiki/Bada_\(operating_system\)](http://en.wikipedia.org/wiki/Bada_(operating_system))

28. <http://en.wikipedia.org/wiki/Tizen>
<https://www.tizen.org/>

29. <https://meego.com/>, <http://en.wikipedia.org/wiki/MeeGo>

30. <http://en.wikipedia.org/wiki/WebOS>

31. http://en.wikipedia.org/wiki/Google_Chrome_OS

32. <http://www.idc.com/getdoc.jsp?containerId=prUS22871611>

33. http://www.symantec.com/content/en/us/about/media/pdfs/symc_mobile_device_security_june2011.pdf

We must compare entire environments:

- Android OS plus the Android Market
- iOS plus Apple's App Store

Adding complexity for Android, many hardware manufacturers take the open-source core provided by Google and make their own modifications, some of which are security related. This fragments the OS space into many proprietary branches maintained in parallel (and usually with delays) with the core OS. Such a situation lengthens the time for OS updates from these manufacturers because fixes to the Android core must be reflected in custom OS versions and be deployed separately. The release of Android Ice Cream Sandwich (AICS) may address this issue, but we cannot judge its effect until AICS becomes the prevalent OS flavor.

Perhaps the key element in the ecosystem of each mobile device is its application store and the source of its software updates.

Application stores

Software distribution for smartphones is significantly different from that for desktops. One key mobile trend is to tie users of a particular vendor to its Internet-based marketplace. This is relatively easy to do because locking customers of mobile phones into a specific GSM/3G network is now accepted. By locking customers of mobile devices into their marketplaces (or app stores), vendors gain a clear financial incentive that can also have a potentially positive effect on security: If most software comes from a single source, this may simplify the vetting of programs and removing of unwanted content.

To commit customers to a marketplace, providers may use digital signing of applications (which is effectively a form of DRM). Otherwise easily shared content (including, but not limited to, software) can affect the providers' ability to avoid piracy. This is the approach taken by Apple. (Only jailbroken iPhones can run programs not signed by Apple.)

Smartphone vendors do have the incentive of calling themselves "the safest provider." That competition gives customers some hope that mobile devices will strive for safety.

With centralized software distribution and a strong drive to provide a safe ecosystem, providers should be able to keep the amount of malware under control by filtering apps submitted to their online stores. (This would be almost impossible to achieve if distribution of software is seriously decentralized.) Nonetheless, the bad guys will always adapt quickly to a new environment.

Reactive versus proactive

The world of mobile software distribution is dominated by Apple's App Store and Google's Android Market. They employ distinctly different policies and apply filtering in separate ways:

- With Apple's rigidly centralized distribution there are only two ways to receive a new app:
 - » From an App Store download
 - » From iOS Mobile Device Management (which requires approval by Apple).³⁴ Any legitimate company can get their software distributed this way.
- Google runs and controls the content in the Android Market. It is very popular but any device can also download applications from third-party markets on the Internet or use a browser to download an APK from a URL.

Apple's store is more tightly controlled and is, at least for now, safer. Android's software distribution is more open and has suffered from malware on multiple occasions—for example, the DroidDream incident in spring 2011, when more than 50 legitimate applications containing malware were distributed.³⁵ Those apps stole information and waited for instructions from a command server (which is typical botnet behavior). DroidDream was the first major Trojan attack against a popular app marketplace. Google had to remove more than 50 malicious apps from the Android Market.

34. <http://www.apple.com/ipad/business/integration/mdm/>

35. <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=399522>

Apple's approach is proactive and focused on prevention. Google's plan is apparently to encourage the creation of apps and deal with the problems as they occur, in a reactive fashion. Google's may be a sensible move to generate a large volume and wide variety of apps, but from the security perspective it creates exactly the kind of environment in which malware gangs feel comfortable.

Clearly, the amount of malware in an online store depends on the quality of filters that decide what is acceptable. (These filters include quality and security checks.) Attackers also take into account the possible return on investment for each environment. This includes the cost (in time and money) of pushing malware through a corresponding filter.

Google's offering is likely to beat Apple's fairly soon in the variety of software. But although customers appreciate a wide choice, they also want access to a safe environment. Achieving these two goals simultaneously is not easy and may require significant investments from the providers to keep their ecosystems clean. We believe that smaller players may have a hard time competing with the large providers because the effort of implementing effective software filters may add too much to their operational costs.

Attacks on Internet marketplaces

Application stores are likely to be manipulated in a manner similar to search-engine abuse, which pushes URLs to the top positions in Internet searches.³⁶ In application stores, the attacks would inflate the reputations of selected applications to drive more users to download them. The same methods could be used both for legitimate software and for malware, which will make it almost impossible to identify attacks on application ranking performed by malware creators.

We have seen attacks in which legitimate (and usually popular) apps were repackaged with malware or advertisements and uploaded again for users to download (sometimes on a different marketplace).³⁷ Such reposting may or may not involve modifying the identity of the app (name/icon/creator). Sometimes the attackers may deliberately keep the original software to suggest it comes from a reputable source—and also to preserve the digital signature and identity of the original author, whose software is now included as an attractive element in a malicious package.

We also expect malware functionality (like data-leakage payloads, which steal users' data) to be embedded into otherwise useful applications. Some of these applications may even be paid for. This would allow malware creators to profit twice—once from charges for the app and once from the stolen data. If the theft is covert and unsuspecting, it may go on for a lengthy period after this dual-purpose software slips through app store filters and while it operates under the radar of security software. For example, dual-purpose software can be programmed to transmit stolen data only if its value is high enough to risk the exposure. For all we know some applications in the App Store might have hidden malicious functionality. This type of attack on the App Store looks like the most probable vector, given that so far Apple has been successful in protecting all but jailbroken iPhones.³⁸

We expect that as the architectures of app stores become more resistant to mass distribution of malware they will still be the victims of targeted attacks. For example, a malware developer could release an application with hidden functionality, entice a user to install that app, and then activate the hidden malicious function. Removing this type of booby-trapped software from app stores would be exceptionally hard.

If malicious attacks on app stores become too hard to mount (very unlikely), then malware gangs could inject malware into applications developed by legitimate companies by infecting their desktops and modifying the software under development. This isn't likely to happen very frequently but if it does it may allow for very wide and rapid malware distribution.

The bad guys often try to stay under the radar by exploiting web vulnerabilities or by distributing malware in small numbers via app stores. These strategies worked well for them in the past, and there is no reason why they should not work in the mobile world. However, these are more complex alternatives to poisoning marketplaces, so we are not likely to see much of this going on until the operators of app stores excel in purifying their markets.

36. http://en.wikipedia.org/wiki/Search_engine_optimization

37. <http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=363542>

38. <http://blogs.mcafee.com/mcafee-labs/get-out-of-jail-not-so-free>

DRM and remote control

All popular mobile OS's require digital signatures on applications, and can restrict and control the distribution of software. (The control is essentially a form of DRM, normally used for restricting the distribution of films and other entertainment.) Google has already exercised its feature of remote application removal built into Android following a malware outbreak—to wipe out already installed Trojans.^{39, 40}

Google can implement multiple restrictions to block bad activities and perform remediation:

- Remove an app from the Android Market. (Google controls only its own store; third-party markets are not covered.)
- Remotely kill already installed apps
- Push a special Android security tool onto the infected devices
- Block Google accounts associated with a bad app

Of course, Google risks making a mistake when implementing any of these measures. Some concerns were voiced about losing a Google account and the data associated with it.⁴¹ A false alarm concerning an application could cause Google to blacklist or remove a person's account associated with this app. This has implications beyond those that are usually associated with false-positives in security software.

As far we know, Apple has not yet had to perform a remote app removal (and given some privacy concerns, a reluctance to use this function would be easy to understand), but we know that iOS does have the same remote-killing capability. It was found by developers of jailbreaking routines for iOS.

Developers

Developers create the software in application stores. Due to the digital signing of apps, they are also part of the ecosystem, tightly “coupled” with the markets.

The reputation and ranking of software (and of the developers) in application stores (as well as in popular external app-ranking web services) are likely to be a prime factor for customers when deciding which applications to install. Reputation will become a commodity—and so it will become a target for malware gangs. Secondary markets that trade reputations will likely emerge, too.

Consequently, we expect to see developers targeted by malicious attacks. One such attack on the Symbian forum was recently reported. Lots of developers' credentials were compromised.⁴² Some developers work on multiple mobile platforms, so this recent breach may have implications beyond Symbian, for example, for Android and iOS, too. Stolen identities (or even the same login credentials on multiple sites) may assist the malware guys in hijacking the reputations of legitimate mobile developers.

Alternatives

The release of HTML5 has the potential to disrupt the app store environment currently dominated by Apple and Google.⁴³ This new HTML standard allows the delivery of content (such as video and audio) through the browser, instead of via specialized apps. HTML5 allows apps to use interface elements that are OS specific, so that they will appear as native borderless apps.

It could make economic sense for some third-party developers to avoid centralized app marketplaces (which demand a fee of all sales) and distribute their content via HTML5-enabled browsers.⁴⁴ This is likely to drive the use of HTML5 instead of custom apps to deliver content.

39. <http://android-developers.blogspot.com/2010/06/exercising-our-remote-application.html>

40. <http://blogs.mcafee.com/enterprise/mobile/google-tool-cleans-up-mobile-malware-dream>

41. <http://www.itworld.com/it-managementstrategy/187543/when-google-kills-your-account-what-happens-your-android-phone>

42. <http://www.infosecurity-us.com/view/20396/nokia-shuts-down-developer-site-after-members-data-was-compromised/>

43. <http://en.wikipedia.org/wiki/HTML5>

44. <http://www.zdnet.com/blog/btl/amazons-cloud-reader-beginning-of-the-html5-surge-vs-apples-app-store-vig/54587?tag=nl.e539>

Changes and predictions

In this chapter we shall look at the current state of malicious software development and how the specificity of mobile devices expands the options available to the bad guys.

Motivated by money

The majority of malware that McAfee Labs sees is inspired by financial gain. There are many ways that malware operations can earn money; mobile devices, unfortunately, add to that long list.

Man-in-the-browser attacks

Stealing or modifying financial information from a browser during an online banking session has been one of the most notable advances in PC malware. Sometimes these Trojans can completely hide malicious transactions from the user. If compromising the browser were as easy on smartphones, then this would undoubtedly be another common attack scenario. However, this is not the case. Both iOS and Android offer strong process isolation, so—unless the OS itself is compromised and the malware runs as root—browser communications should be neither visible nor modifiable by any other program.

Multiplatform threats (Zeus and SpyEye)

Mobile phones now sometimes provide additional security for online banking.⁴⁵ They are used in two ways:

- To provide additional login information (dual authentication)
- To inform customers (via SMS, for example) about transactions or account balances

In 2010, the Zitmo Trojan for Android mounted a successful attack on dual authentication.⁴⁶ Zitmo, “Zeus in the Mobile,” is an extension of the PC-based Zeus banking-Trojan family.

The Spitmo (“SpyEye in the Mobile”) Trojan for Android appeared in September 2011 and also attacks dual authentication. It is a part of another large family of PC banking Trojans.⁴⁷

Premium numbers and SMS

One of the simplest ways to generate profit from malware attacks on mobile devices is to place a call or send short message service (SMS) texts to pay-for premium numbers. If this activity is infrequent (for example, only once a week during the night) and concealed (by erasing the logs and using the hiding capabilities of rootkits), then it may go unnoticed for a long time.

Tracking such activity would seem fairly simple, as providers should have a good record of companies using premium numbers and there should always be records of financial transactions. In reality, this is seldom a trivial task, as calls and texts may go abroad. Some countries are more relaxed regarding abuse in this area, especially if the activity has ceased (which is common). As we mentioned, in countries such as Russia malicious SMS’s were exceptionally common on J2ME. The key property to this malware’s popularity is the software’s ability to covertly send messages. In this case, Android is more risky than iOS because in Android permissions are assigned once at installation and cannot be dynamically controlled. (An external audit of permission with a tool like App Alert is one way to keep an eye on this functionality.)⁴⁸

Microsoft Windows 8 is supposed to include support for SMS as well as for mobile devices. We do not have full details yet, but it is conceivable that malware on a PC might be able to send an SMS (or make a call?) via an attached Bluetooth phone. We plan to research this as soon as possible.

Near-Field Communications

Mobile devices require special hardware to support near-field communications (NFC) payments. The range of NFC is limited to just a few centimeters.

45. <http://blogs.mcafee.com/mcafee-labs/mobile-reunion-hackers-and-banks>

46. <http://blogs.mcafee.com/mcafee-labs/dissecting-zeus-for-android-or-is-it-just-an-sms-spyware>,
<http://home.mcafee.com/VirusInfo/VirusProfile.aspx?key=290717>

47. <http://blogs.mcafee.com/mcafee-labs/spitmo-vs-zitmo-banking-trojans-target-android>

48. <https://market.android.com/details?id=com.mcafee.mobile.privacy>

There are two standard methods of preventing NFC abuses:

- Limiting the size of transactions. (The average NFC transaction in the United Kingdom is less than £5, which is less than US\$8 at current rates.)
- Blocking NFC-only profiles to prevent stolen cards from being used without PINs, signatures, or any identification

However, thieves may develop special equipment to boost the range of NFC in crowded areas (airports, transport, theaters, cinemas, stadiums, etc.) and by using short-lived merchant accounts specifically for skimming purposes. We are certain to see such attacks as soon as enough devices are ready for NFC payments, credit cards, electronic wallets, and Bitcoins.

Many phones carry financial data (either directly or in a form of saved login details for bank websites). When e-wallets become more common, thieves will certainly look for ways to empty them. The level of security risks associated with e-wallets should be at least comparable with PIN-enabled credit cards, perhaps even worse due to additional attacks via device software.

Android apps that carry Bitcoins, a form of online money, already exist, and this type of virtual currency can be used, for example, in NFC transactions.⁴⁹ Bitcoins transactions based on quick-response (QR) codes are also possible.⁵⁰

The loss of financial data or currency could be catastrophic for their owners. Therefore, mobile devices should use strict data-loss prevention (DLP) methods to limit the risks. (DLP software is normally used as a safety net to block the loss of critical data if a computer system is compromised despite other protective measures. In this sense it is a damage-limiting tool.)

Some data (such as corporate secrets) may be very valuable for attackers but has less liquidity—as the thieves must find the right buyer and this may be quite risky. This danger makes stealing Bitcoins particularly attractive because this type of asset is very hard to trace. Even traditional “mules” would not be required to access the funds obtained from stealing Bitcoins. If this virtual currency becomes common, then we should expect a particular focus from the bad guys to steal it.

Connectivity

Defeating the security of a smartphone’s connections gives malware manufacturers a lot of options. Let’s look closer at the connectivity spectrum of mobile devices.

Femtocells

Femtocells are a cheap way to extend mobile network coverage by setting up a small mobile phone base station with an access range of about 10 to 30 meters.⁵¹ They can be used in homes or small offices. Femtocells connect to the mobile operator infrastructure via the Internet and are typically small, ready-to-use devices with an Ethernet connector and a power plug. Femtocells are very attractive for mobile operators, as they can provide coverage where signals from large towers are weak and where building additional towers is not economical. Moreover, by selling femtocells operators may generate additional revenue.

Other cell types and their ranges:

- Microcell: less than two kilometers
- Picocell: less than 200 meters
- Femtocell: about 10 meters (AT&T calls its product in this range a microcell)

Unfortunately, there are significant security risks associated with femtocells. Essentially, they are small mobile towers that can be acquired (for US\$100–200) and reverse-engineered. They talk to the operator infrastructure via the Internet (TCP/IP), and unless they are very robust they can be abused in many ways. The most obvious attack is to install a hacked femtocell close to a home or office and start an MITM attack on all mobile communications. This would allow full access to voice and SMS for all devices that connect to this rogue cell.

Some operators have decided not to offer femtocells to their customers, presumably to avoid security risks. During the BlackHat 2011 conference we saw multiple weaknesses demonstrated in a particular femtocell model (running proprietary embedded Linux and costing €99 in France).⁵² One problem with this model was that a factory reset wipes the local configuration and then reconfigures the device from scratch. Resets could be performed multiple times, allowing a comprehensive analysis of the data exchange during initialization and, ultimately, the reverse-engineering of the setup protocol. For example, this model exchanges certificates with an operator's infrastructure but does not check the authenticity of the operator. Plus, the key for future encryption is transmitted in plaintext.

Femtocells are frequently remotely managed with the help of the TR-069 protocol; this service may also be vulnerable.⁵³

A general problem with embedded devices such as femtocells is that they usually run as root—so any successful exploit gains full access to the device. (There's no need to look for additional vulnerabilities to elevate the privilege level.) Instructions for how to break into certain models of femtocells can be found online, and implementation weaknesses are common.

Femtocells also have privacy issues. For example, they may keep a list of nearby cell towers (for location info). They also keep the subscriber's ID (IMSI) and phone ID (IMEI). Some femtocells provide a web page for configuration. If that web access is open, weak, or cracked, then the implications can be severe. Another privacy concern reported at BlackHat 2011 was that a particular femtocell model aggregates the logs and periodically sends the unencrypted package to the operator via FTP. Among other data, these logs include information on each user's activity.

Storage devices

Storage hardware provides additional propagation vectors for malware. For example, an SD card with a program or an installation package can be easily inserted into most devices and left there (perhaps when the owner is momentarily distracted). It may take a while before some of the programs from that card execute (or files containing exploits are opened). This vector would be ideal for targeted malware. Stolen data can later be retrieved by removing the SD card.

We recommend tightening the physical security of SD cards (making them less easy to insert or eject)—perhaps even putting them under OS control to prevent media manipulation.

Some SD cards permanently serve to increase the memory space on a device. They act like a hard drive and should be appropriately protected (perhaps by the total encryption of card content using modern cryptographic methods).

Android phones would certainly benefit from functionality supporting access control and the encryption of applications data files on SD cards (or any other removable media).

Another danger to mobile storage (internal storage, SD card, or both) arises when it is exposed to a PC. When a mobile device is connected as a "slave" (usually via a micro-USB connector) to a laptop or desktop, its internal storage can be mapped by Windows as an external drive. This opens the device to some types of attacks (such as AutoRun or the LNK vulnerability used by the Stuxnet worm) that can automatically execute malware on a PC. The danger is present with a cable connection or by plugging the SD card into the PC. If the attackers can't auto-execute malware, they can simply place a malicious PC program onto an SD card and hope it will eventually get executed on a PC. This vector expands the scope of a compromise from one mobile device.

Bluetooth

Recent demonstrations show that the Bluetooth network stack can be exploited to allow remote code execution and the control of motor vehicles without any physical connections.⁵⁴ It is equally possible that attacks on mobile phones could be launched via Bluetooth. An attack requires proximity, but the propagation of the Cabir worm on Symbian phones that occurred when mobile users were in a stadium confirms that this factor is perhaps not as limiting as one would expect. A Bluetooth stack vulnerability (or even simple MAC spoofing) would be the most popular vector to launch a targeted attack: Due to the proximity it is very easy to make sure that only a selected person (or a group) will be exposed to the attack. This would be perfect for a targeted APT.

Bridging networks

Mobile devices are unique in their promiscuity. They are usually permanently connected to at least one mobile operator tower (via GSM, 3G, etc.) but they also connect to local networks (usually via Wi-Fi) and in close range (via Femtocell, Bluetooth, and infrared). Due to their mobility, most devices change many of these connections during a day. This makes the job for attackers easier, as they have a wider choice of entry points and protocols. They prefer to go for the weakest link—frequently the home connection.

There is also the issue of reconnections. The ability for attackers to repeatedly “sniff” network traffic during reconnection (for example, when a person comes home from work the mobile phone may be reconnecting to the same Femtocell) may allow them to mount attacks that they would otherwise be unable to consider.

Naturally, if any of the networks is compromised, then the mobile device may be compromised. When that device reconnects to a corporate network, it may become an entry point for further malicious actions.

Abusing security functions

Backups and data encryption

Many mobile devices have backups integrated into the lifecycle of the device. For example, iPhones sync with a computer. Android devices save data online linked to a Google account. We've seen reports of breaking iOS encryption, but physical access to the device is required.⁵⁵ (Using this method cannot decrypt a backup from a PC that an iPhone syncs to.)

However, there are always security risks associated with the backup process:

- Data being backed up remotely may not be properly protected in transit. The history of Apple's modifications to securing the syncing process shows that each OS update strengthens security. But that means there were some weaknesses that were gradually identified and addressed.⁵⁶
- Backup data is a valuable target for cybercriminals. iPhone backups are encrypted; for Android there are several third-party apps to do that.
- Certain types of attacks may occur even when the data is properly encrypted. For example, measuring the time required for a backup (or the size of the backup file) gives an estimate of how heavily a device is used, which may indicate to cybercriminals whether the phone is a worthy target.
- Mobile devices that use custom backup systems bear additional risks, as these systems attract less scrutiny from the research community
- Triggering a remote backup (or a restore) on many devices can cause a DoS scenario, so this capability must be appropriately protected

54. <http://www.technologyreview.com/computing/35094>

55. <http://www.itproportal.com/2011/05/25/russian-security-group-breaks-ios-encryption-says-few-groups-capable-repeating-steps>

56. D.D.Zovi, "Apple iOS Security Evaluation: Vulnerability Analysis and Data Encryption." BlackHat 2011, Las Vegas

Remote wipe

It is a common function, sometimes provided by add-on software, to do a remote wipe if the device is lost or stolen. This is frequently used in corporate environments whose policies are managed by IT teams. (An IT policy can wipe a BlackBerry, for example.).⁵⁷ This function is useful to protect sensitive information but, if abused, can be very dangerous.

IT administrators who can remotely trigger a wipe should follow stringent policies for this action. Remote backup and remote wipe capabilities should be designed in a way that requires physical access to the equipment initiating these functions. If remote triggering is allowed, then the risks can be very high.

Storing passwords

Mobile devices are frequently used to store sensitive data. Nokia phones (even before smartphones) had a wallet function that could store sensitive data (login/password pairs, credit card numbers, PINs, bank details, passphrases, etc.).

Mobile devices store data in apps, which likely makes the data more vulnerable. There are several potential loopholes:

- If there is no proper isolation between applications, then sensitive data may be exposed to other programs
- If encryption is poor, it may be possible to decrypt and subsequently transmit the data. There is, for example, a vulnerability in one version of Skype for Android that stores private data in a SQLite database without encryption.⁵⁸

To increase usability, some password aggregators operate in active mode—supplying login data automatically, which means less control over when sensitive data is decrypted (and less user control over that data).

Locking the device

Mobile devices need to prevent unauthorized access in case they are lost or misplaced, even temporarily. The most common solution is to lock a device after a short period of inactivity and require a PIN or password to unlock it.

This step provides rudimentary security but it is far from a perfect solution. PINs are easy to capture by simple shoulder-surfing.

Sometimes sensitive applications use a second password—a good security practice—but unless this is enforced by a security policy users may turn off the second password to increase convenience.

Standard security policy is to use a 4-digit PIN (iPhone) or pattern recognition (Android). This flimsy defense needs to be changed for a more secure alternative (at least a longer alphanumeric string), especially for the phones used for business purposes.

A very high percentage of owners lock their devices with a short PIN.⁵⁹ Many are unaware that alternatives exist (for example, a “non-simple” security option on the iPhone). Moreover, we tend to pick very weak PINs. (About 11 percent of PINs are one of five combinations: “1234,” “0000,” “1111,” “2580,” or “0852.”⁶⁰

It is a pity that fingerprint scanning and iris recognition are not yet among the login options.

57. http://docs.blackberry.com/en/admin/deliverables/4222/Remote_Wipe_Reset_to_Factory_Defaults_250402_11.jsp

58. <http://www.androidpolice.com/2011/04/14/exclusive-vulnerability-in-skype-for-android-is-exposing-your-name-phone-number-chat-logs-and-a-lot-more>

59. D.D.Zovi, “Apple iOS Security Evaluation: Vulnerability Analysis and Data Encryption.” BlackHat 2011, Las Vegas

60. <http://www.gadgetnew.info/iphone-top-10-pin-codes-picked-by-users>

Multiple identities

Many mobile devices are not truly personal devices; they are shared, for example, by family members, friends and relatives. Sharing introduces additional security risks:

- A wider variety of applications on the device
- Personal data for multiple individuals
- Less control over paid services
- A higher probability of misplacing the device
- Exposing children to undesirable content

Security measures that restrict sharing, especially for business devices, would be highly useful. Such measures may include:

- A whitelist for allowed applications
- Limiting capabilities for multiuser applications (for example, allowing only a single identity in email)
- Locking business data (strong encryption) when outside of the business environment
- Parental controls software

Consumerization of IT

Many home devices appear in enterprises and connect to the networks. Employees want to use their mobile devices for business, while IT administrators prefer to trust systems that they have built and control. The pressure to adopt non-IT devices in work environment is called the consumerization of IT. This process is not entirely safe: Some people may be carrying malware-controlled, Internet-connected devices capable of video and sound recording without their knowledge.⁶¹

We saw similar problems (but on a smaller scale) when the first personal laptops entered corporate environments. With smartphones, however, the problem is a lot more common.

It is inevitable that mobile devices will start adopting management functions that allow IT personnel to enforce certain security policies. Phones will have DLP functions, which will likely be delivered either as part of the OS or in security software. For DLP to be effective, IT administrators must manage the rules very carefully.

Man in the middle attacks

The multiple connections of mobile devices and their frequent changes increase the chances of MITM attacks, so we expect these to grow. This danger may be exacerbated by screen-size limitations and, for example, the lack of a secure sockets layer (SSL) indicator in the browser. Software developers, attempting to pack more information into a small screen, might leave some (or all) security information off it.

The SSL functionality itself may be lacking or exploitable, as we've seen in one case.⁶² Trusted certificate providers have been compromised (Comodo and more recently DigiNotar).⁶³ When a common and trusted SSL certificate is hacked, an MITM attack could occur, leading to the theft of data (such as login credentials) or manipulation of data (returning incorrect instead of true data). Attacks similar to SSL-based MITM can happen when the Domain Name System (DNS) is compromised.

Vulnerabilities in applications

Mobile malware doesn't live only in app stores. Just as on a PC, a smartphone visiting a malicious or compromised website can pick up malware via an application exploit. The attacker needs access to a common vulnerability in some popular app.

61. http://droidsecurity.appspot.com/securitycenter/securitypost_20110804.htm

62. <https://www.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt>

63. http://www.computerworld.com/s/article/9218676/Sniffer_hijacks_secure_traffic_from_unpatched_iPhones
<https://community.mcafee.com/thread/38704?tstart=0>

Once attackers have access to certain vulnerabilities, they need to perform a two-stage attack:

- Run unsigned code when a user visits a website or opens a document. Vulnerabilities in browsers (Safari, Opera, etc.) or in PDF/DOC/HTML software can allow this type of code injection and execution. (For example, the latest jailbreaking exploit for iOS employs a vulnerability in PDF font handling.)
- Modify the OS kernel to allow malware to execute with the highest possible privilege. (This requires an exploitation of an elevation-of-privilege vulnerability.) At this point malware authors normally turn off updates of OS or AV software to ensure a long run.

It is a misconception that nonjailbroken devices such as iPhones are immune to malware because all the software comes from a trusted source (Apple's App Store). This is not correct. First, malware in the App Store is unlikely but not impossible. Second, and more important, there are vulnerabilities in devices that allow the escalation of privileges and running of unsigned code in the kernel. If there were no such exploits, then jailbreaking would be impossible. The mere fact that all iOS versions (up to 4.3.3) have been jailbroken proves that such deep vulnerabilities do exist.

Replicating malware

Parasitic viruses

Most mobile OS's require that each app be signed. This makes parasitic viruses impossible because attaching to a host requires modifying it, and this will break a signature. There is a possibility, however, of a parasitic virus spreading on OS's with crippled security (such as jailbroken iPhones). In any case, parasitic viruses would be pretty pointless for malware writers, as users no longer share apps directly between devices.

The only type of parasitic virus that could survive in this environment is one which carries its own source code and when executed adds itself in source-code form to the source code of other apps. This danger is not as impossible as it sounds; there is a field virus like this for Windows: W32/Induc.⁶⁴ Still, for a virus to propagate, it must be able to hop from a mobile device into a source-control system where apps are developed. The most likely scenario would be with weak network security (for example, an open SMB share in a Wi-Fi network that contains an app development repository) so that a virus on a phone could modify the source code of the apps on desktops and include a copy of itself. However, given that malware on mobile phones is unlikely to have access to app development repositories, we will gladly classify this threat as exceptionally small.

More likely scenarios are backdoor Trojans or botnets that provide unauthorized access to mobile-software source control and build systems on traditional desktops. In these cases attackers could make modifications to apps under development and "poison" the sources.

Worms

A worm is malware that doesn't change, so if it is digitally signed this signature will always stay valid. That makes this type of malware simpler compared with parasitic viruses. The method of infecting app sources that we discussed for parasitic viruses would also work for worms. However, a worm can skip the most complex step—modifying app sources. Instead, if the worm finds unprotected app store credentials, it can immediately upload a copy of itself (or multiple copies, perhaps under different names) to an app store for other users to download.

This scenario is also not very likely because finding valid app store credentials should be a rare event. So, unless a worm has a precompiled list of credentials (embedded into it or remotely accessible) it would have troubles replicating.

Autoworms

It is possible that a vulnerability will be discovered that allows remote code execution on some popular strain of mobile devices. If such a flaw appeared in any network protocol used by any common application, then it may allow an autoworm to propagate from one device to another. (By autoworm we mean a virus—like W32/CodeRed or W32/Slammer—that propagates without human intervention.)

Operators and security companies should be prepared for such an event because it could cause explosive traffic (unless the worm itself has some throttling capability) and has a potential to seriously disrupt mobile communications.

Vulnerabilities are usually very sensitive to the targeted environment and cannot operate on multiple OS's and hardware. Therefore, it is unlikely that an autopropagating worm would affect more than one type of mobile device. However, with the increasing standardization in smartphones some attacks spanning multiple hardware types might emerge (such as a vulnerability in multiple devices running Android).

Poisoning and drive bys

Applications rarely move from one device to another. We expect attacks will focus on poisoning application stores and inflating the popularity of malware placed there. Attacks that manipulate the ratings of applications in app stores (similar to search-engine optimization attacks on Google and other search engine providers) are likely to grow rapidly.

Once a piece of malware is implanted in the app store and achieves an acceptable rating (reputation) level there's probably no reason to circulate a link (for example, by sending it in a spam campaign or social network). The store should provide a constant flow of victims.

Deploying malware when a user visits a web page (known as drive-by downloads) is a common way to infect PCs by exploiting vulnerabilities in browsers or their plug-ins. The effectiveness of similar attacks on mobile phones would also depend on vulnerabilities in the browsers and Internet-facing applications (which open frequently shared files such as PDF, JPG, DOC, XLS, PPT, HTML, etc.). If the number of vulnerabilities in these applications does not diminish, then we can expect the frequency of attacks using malicious URLs to grow.

Phishing and whaling

Phishing attacks against mobile devices have already occurred and we're sure to see more.⁶⁵ Phishing is effectively a social engineering attack and the methods are similar whether the victims are on desktops or smartphones.

Whaling is a form of phishing that attacks high-profile targets (for example, CEOs, CFOs, and CTOs). It may be used in multistage targeted attacks. If the first stage of whaling succeeds, then the attack could expand in multiple directions within an organization. Installing an APT is perhaps the most efficient way to continuously steal sensitive data.

APTs and rootkits

Advanced persistent threats are designed to hide their presence (hence "advanced"). The technology most frequently used is a rootkit (a form of stealth malware deployment).

Rootkits have been a growing problem on Windows, despite improvements to the OS. At DefCon in 2010 we saw that implementing a rootkit on Android is relatively straightforward.⁶⁶ That presentation also nicely demonstrated how the rootkit was very sensitive to the Android OS version. It remains a complex task to create a rootkit that can successfully attack multiple versions of Android.

Apart from solving the "portability" problem to allow multiversion rootkit operations, malware developers want to be able to deploy a rootkit remotely (without physical control of the device). To do that they must remotely execute malicious code with root privileges, which in practical terms requires a remote code-execution exploit and an elevation-of-privilege exploit. An attacker has to have both exploits (unless the former is in the kernel and immediately allows code execution with root privilege)—otherwise remote deployment is not possible.

A rootkit could also be installed on a smartphone before the user receives it (at the factory or in the shop). This would be optimal for an attacker, who would have access, exact specifications, and the OS configuration—no portability or remote-deployment problems. The best bet, however, for surreptitiously installing an APT is to find a poorly locked smartphone and "borrow" the device.

65. <http://www.malwarecity.com/blog/mobile-phishing-do-you-know-where-that-link-leads-to-1021.html>

66. <https://www.defcon.org/images/defcon-18/dc-18-presentations/Trustwave-Spiderlabs/DEFCON-18-Trustwave-Spiderlabs-Android-Rootkit-WP.pdf>

For popular OS's patching known exploits takes usually less than two weeks (perhaps a bit longer for devices with customized versions of Android). So the attackers have a limited window of vulnerability. Google claims that the open-source nature of Android allows programmers and security experts to find bugs more quickly, ultimately leading to more secure code. This approach seems to be working—the total number of reported vulnerabilities in Android is indeed lower than in iOS. On the other hand, the open source may make it simpler to find very deep security problems and exploit them. Some of these exploitable bugs would be impossible to find without consulting the source.

The danger is limited, though, because zero-day exploits are hard to find and are expensive to purchase. We also believe that with the progressive hardening of the OS's the frequency of zero-day exploits will decline (and jailbreaking will get progressively harder). At the same time, black-market prices for zero days will inevitably go up. This dynamic should push many exploits out of the reach of most malware authors.

Malware authors might be priced out of the market for zero-days, but this might not limit some APT development, which appears to be funded by governments (as was strongly suggested by the W32/Stuxnet case).

Patching and updating apps on mobile phones is quicker than on desktops. The mechanisms for this are built into the OS's, and most devices are almost permanently connected to the Internet. These factors will ensure that old (unpatched) mobile versions are updated more quickly than is the case for desktop computers. Quicker patching and updating diminishes the window of exposure.

Virtual OS

The ultimate rootkit would convince the OS that it was running on genuine hardware—while the rootkit was instead intercepting all OS-hardware interaction and tampering with all data. This type of attack based on virtualization is called a blue pill.⁶⁷ Rather than a man in the middle, it represents a “man in the hardware” attack.

The current generation of mobile devices does not yet embrace the benefits of virtualization (likely due to the restrictions of low power consumption). But the use of virtualization in smartphones will inevitably grow (for example, to run multiple OS's on the same device) and some attacks will undoubtedly emerge, though we doubt this will happen for several years.

That's entertainment

When mobile devices are not used for business they are often used for entertainment. The same device frequently serves both functions.

In general terms entertainment data (films and music) is similar to corporate data—both need to be protected. DRM and DLP are the most common security technologies used for that. We expect mobile hardware to keep the data safe (from storage to screen, and from the input devices to the app or provider) and to bind the data to a specific device. Meanwhile threats to mobile hardware (including reflashing microcode for components or replacing them) will inevitably increase. Attackers will try to break the one-to-one binding. It is easier to perform this type of action on mobile devices because physical access will sometimes be available.

We would like to see tamper-proof devices that will automatically wipe the data if they are opened or if they detect invalid attempts to reprogram or replace hardware. This is tricky in a battery-powered device; it can detect nothing if the battery is removed first. (Is it perhaps for security reasons that Apple does not make the iPhone battery user accessible? Even if this was not the intention, we think it is a good idea!)

The most sensitive mobile devices could employ proximity sensors, for example, with the capability to securely lock (or wipe) the device if it moves too far from the owner.

Mobile fake alerts

Fake AV software is one of the most common and visible types of malware for Windows machines. This social engineering scam is also called scareware, rogue AV, and fake-alert software. The malware produces false security warnings designed to scare the users into paying to “fix” these “problems.” (Fake AV claims to run a scan for free and then usually demands payment for the “full” version, which “cleans” the system.)

We expect Fake AV to show a lot of interest in mobile devices because it’s been a big money maker on PCs. We’ve already read reports of several cases: one on J2ME⁶⁸ and another, in April 2011, in which Android malware pretended to be a security tool.⁶⁹

Social networks

Connecting to social networks (Twitter, Facebook, LinkedIn, and others) from mobile devices is gaining popularity as on-the-go access is well suited for these resources. At the same time, lots of information on social network servers is marked private. But even when the security of the social network works properly, connecting via a compromised mobile device could allow data to be stolen.

We particularly worry about free Wi-Fi networks (which most people use instead of paid-for alternatives) that can easily sniff unencrypted traffic and grab passwords (which is very easy unless an SSL connection is used). You can reach facebook.com, for example, with either HTTPS or HTTP, but many are unaware of the former.

Remote control

Mobile devices are increasingly used to remotely manage resources⁷⁰ or provide full remote access to PCs.⁷¹ Tablet devices are also very convenient for light administration and control duties. (Tablets offer screens almost as big as laptops and periodic checks on corporate assets or processes rarely require extensive text input, so the lack of a physical keyboard and mouse is frequently no drawback.)

However, if the mobile device is compromised, then attempts to penetrate the corporate network (for example, by exposing an internal server to public access, which could leak corporate secrets to Anonymous or LulzSec groups) or to manipulate controlled processes (stopping a manufacturing line) could be much easier. An insecure mobile device could open a door for subversion, sabotage, and even acts of terrorism or cyberwarfare.

Industrial control

After the spectacular attack of the W32/Stuxnet worm on the uranium-enrichment infrastructure in Iran, we have to wonder about the safety of industrial systems using mobile networks for communication, reporting, or alerting. There are several ways in which mobile communication can be used in SCADA industrial-control systems:⁷²

- Data acquisition (in read-only mode): gathering inputs from thermometers, door status, etc.
- Control (write mode): when the SCADA system, for example, opens electronic locks, shuts valves, or controls the speed of motors or centrifuges
- Alerting in emergencies or other cases: sending an SMS or email, for example, if the temperature in a greenhouse falls below a certain level

Remote units are fairly frequently connected to the central controller of SCADA systems via mobile networks. Such systems are widely available⁷³ and also widely used (if you Google “SCADA+GSM+RTU,” you will get nearly a million hits), so the problem is real.

68. http://www.securelist.com/en/blog/208187561/Antivirus_Fraudware_Goes_Mobile

69. <http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=TrojanSpy%3AAndroidOS%2FLanucher.A>

70. <http://itunes.apple.com/us/app/vmware-vsphere-client-for/id417323354?mt=8>

71. <http://itunes.apple.com/us/app/remote-desktop-lite-rdp/id288362576>

72. <http://en.wikipedia.org/wiki/SCADA>

73. <http://www.ff-automation.com/products/gsm-rtu.shtml>

Mobile phones are often used for emergency reporting. Most industry and IT monitoring tools plus security alarms for business and home use can send alerts via SMS or voice calls using recorded or synthesized voice messages. Blocking or generating forged alerts may become part of future attacks. Any distributed SCADA deployment should include a focused security analysis of mobile devices.

In general, attacks on distributed SCADA systems may involve:

- Disrupting communications (DoS attack)
- Stealing data (passive MITM)
- Manipulating data (active MITM)

Manipulating data may give attackers full control over remote processes as well as what is reported to the control unit. This access will provide completely stealthy operations that are invisible to central controls.

Botnets

Mobile devices can serve botnets as easily as Windows computers do. Some mobile malware has already exhibited botnet properties.⁷⁴

Botnets may cause some specific problems when employing mobile devices. There may be effects on the network operators:

- Increased bandwidth outside of the TCP/IP protocol (general packet radio service, 3G)
- Mobile bots may attack the operator's infrastructure. A portion of a botnet in a certain area may initiate restarts or reconnects, which is an expensive operation and can cause a DoS for a tower, multiple towers, or even the entire provider. Given the maximum range of phones at around 20-25 kilometers, it may be possible to command a sufficient number of bots within the range of one tower (especially in a populated area) to "crash" it.

In some cases a botnet may attack a specific person or a phone number by flooding it with SMS's or voice calls.

Due to their wide choice of hardware, botnets can initiate more types of attacks (voice, video, GPS) and serve as launch pads for infections of other computers via any connection (PC, Wi-Fi, Bluetooth, SD card, USB, etc.).

Conclusion

The security of OS's for mobile devices is advancing rapidly and that makes certain types of malware (parasitic viruses, for example) nearly impossible. However, due to the promiscuous connectivity of smartphones and, most important, the inherent problems related to the filtering of malware in app stores, we expect mobile malware to increase.

We anticipate massive attacks on app stores that will start with simple malware posts and gradually involve manipulating developers' reputations and infiltrating app source-control systems. Most malware will still focus on financial gains, from NFC transactions to Bitcoins to e-wallets.

The advanced hardware capabilities (voice, camera, GPS) of most smartphones should increase the average cost of compromises because attackers can steal more sensitive information.

We expect MITM attacks (based on SSL and DNS compromises, Femtocells, etc.) to grow—with an emphasis on deploying APT/rootkit malware in corporate and industrial areas and exploiting zero-day vulnerabilities.

Security features in OS's (especially Android) will have to improve along with the physical security of mobile devices.

Acknowledgements

I am very grateful to my colleagues Carlos Castillo, Michael Price, and Jimmy Shah for their valuable input and comments.

About the Author

Dr. Igor Muttik is Principal Architect for McAfee Labs. He holds a Ph.D. in physics and mathematics. Muttik's studies of the first computer viruses led to his career at Dr. Solomon's Software, which was later acquired by McAfee. In addition to his research work on malware, he is a frequent presenter at security conferences around the world.

About McAfee Labs

McAfee Labs is the global research team of McAfee. With the only research organization devoted to all threat vectors—malware, web, email, network, and vulnerabilities—McAfee Labs gathers intelligence from its millions of sensors and its cloud-based service McAfee Global Threat Intelligence™. The McAfee Labs team of 350 multidisciplinary researchers in 30 countries follows the complete range of threats in real time, identifying application vulnerabilities, analyzing and correlating risks, and enabling instant remediation to protect enterprises and the public.

About McAfee

McAfee, a wholly owned subsidiary of Intel Corporation (NASDAQ:INTC), is the world's largest dedicated security technology company. McAfee delivers proactive and proven solutions and services that help secure systems, networks, and mobile devices around the world, allowing users to safely connect to the Internet, browse, and shop the web more securely. Backed by its unrivaled Global Threat Intelligence, McAfee creates innovative products that empower home users, businesses, the public sector, and service providers by enabling them to prove compliance with regulations, protect data, prevent disruptions, identify vulnerabilities, and continuously monitor and improve their security. McAfee is relentlessly focused on finding new ways to keep our customers safe. www.mcafee.com

