- Do email and DNS servers function flawlessly? By using email round-trip, DNS sensors and specific email sensors, you can constantly check for correct functionality of those services.

These questions cannot be answered with external network monitoring. Monitoring software must be run directly in the private cloud, or the monitoring tool must offer the possibility to monitor the cloud using remote probes. Such a probe can monitor the following parameters – for example, for each virtual server that runs in the private cloud, as well as for the host servers:

- CPU usage.
- Memory usage (page files, swap file, page faults, etc).
- Network traffic.
- Hard drive access, free disk space and read/write times during disk access.
- Low-level system parameters (eg, length of processor queue, context switches).
- Web server's HTTP response time.

Critical processes such as SQL or web servers are often monitored individually, in particular for CPU and memory usage. In addition, the firewall condition (bandwidth use, CPU) can be monitored. If one of these measured variables lies outside of a defined range (eg, CPU usage over 95% for more than two or five minutes), the monitoring solution will send notifications to the administrator.

Conclusion

With the increasing use of cloud computing, system administrators are facing new challenges and new security risks. Choosing a private cloud, rather than a public cloud, allows network administrators more control over the system and a professional network monitoring solution will enable them to make their networks more secure – directly and indirectly.

The IT department must look into the capacity requirements of each application in the planning stages of the cloud in order to calculate resources to meet the demand. The connection to the cloud must be extensively monitored, as it is imperative that the user has constant access to all applications during operation. At the same time, smooth operation of all systems and connections within the private cloud must be guaranteed. A network monitoring solution should therefore monitor all services and resources from every perspective. This ensures continuous system availability, and capacity overloads can be systematically avoided through long-term planning based on extensive monitoring data.

About the author:

Dirk Paessler is the founder and CEO of Paessler, which provides network monitoring and testing solutions. For more information, visit www.paessler.com.

References

1. 'The Corporate IT Forum Cloud Computing Reality Checker'. The Corporate IT Forum, January 2012. Accessed Nov 2012. www.corporateitforum.com/experience-hub/reality-checkers/doc_details/5304-cloud-computing-in-2012.

Android malware and mitigations

Steve Mansfield-Devine, editor, Network Security

Steve Mansfield-Devine

Android might be a victim of its own popularity. Just as the ubiquitous nature of Windows made it an enticing target for malware writers and cyber-criminals, so it is with Android. Google's mobile OS is on more smartphones than any other operating system and products such as the Nexus 7 and Nexus 10 may finally result in Android gaining real traction in the tablet market. IDC estimates that more than 100 million Android smartphones were sold just in the first quarter of 2012, and in the third quarter, three out of four smartphones sold were running the OS. The maliciously inclined have not been slow to exploit this opportunity, but are they being helped by technical and procedural flaws in Android's architecture and the way apps are distributed?

In the two previous articles in this series, we examined some of the technical issues. In this final article, we'll take a look at the nature of the malware problem and how Google's open approach to distribution makes implementing countermeasures difficult. But mitigation is possible, and there are signs that something is finally being done about the problem.

Software issues

It seems that Android may have reached some kind of tipping point. The platform has been around for years, but its exploitation as a malware target has ramped up considerably in recent months. Figures for Android malware vary widely depending on whom you ask. Anti-malware vendors are usually dependent for their estimates on infections reported back by their software installed on customers' devices. However, as take-up of anti-malware on portable devices is very low, the

reliability of such figures is open to debate. There are also variations in the way that researchers categorise malware families. That said, the one thing on which pretty much everyone agrees is that Android malware is increasing – rapidly.

For example, the latest report (at the time of writing) from TrendLabs claims that the number of "malicious and potentially dangerous and high-risk" apps the firm had seen jumped from 30,000 in June 2012 to 175,000 just three months later.[1] Of course, the definition used by the company is somewhat broad: an app could be designated as 'high-risk' if it has the potential to leak personal information accidentally, regardless of whether such a flaw is actually being exploited. The business of anti-malware firms is to be pessimistic and treat as a problem even the most theoretical of threats. Yet this nevertheless highlights the fact that problems are endemic on the platform. And if the Windows world has taught us anything, it's that any flaw is likely to be exploited eventually.

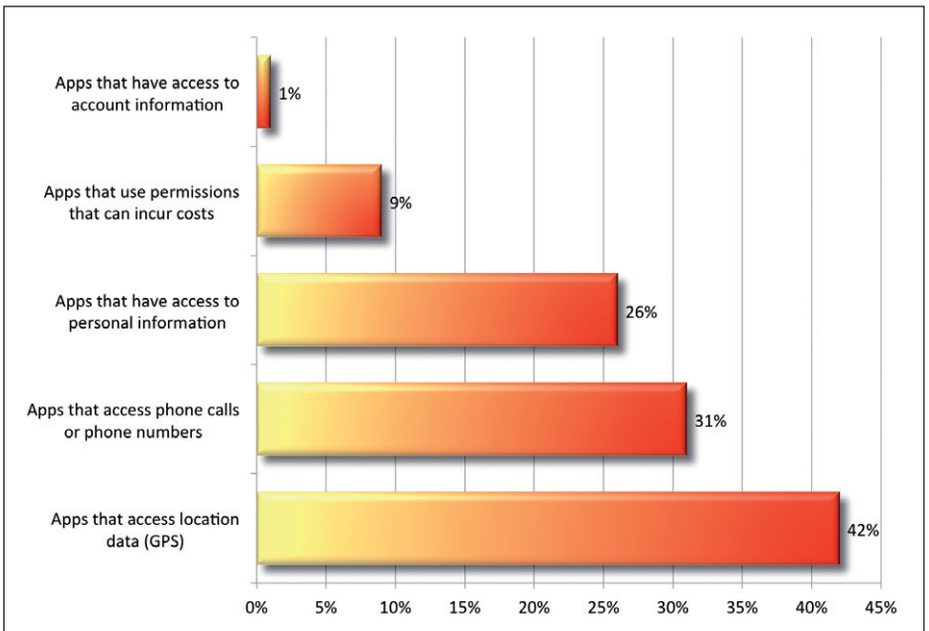

Figure 1: Percentage of apps analysed by Bit9 that request access to potentially sensitive data.

Developer problems

A large number of the problems with Android stem not directly from malicious activity – in the form of trojanised apps or 'traditional' malware – but from poor programming practices that can lead to vulnerable data. OWASP characterises the top 10 mobile risks as:[2]

- Insecure data storage
- Weak server-side controls
- Insufficient Transport Layer protection
- Client-side injection
- Poor authorisation and authentication
- Improper session handling
- Security decisions via untrusted inputs
- Side channel data leakage
- Broken cryptography
- Sensitive information disclosure.

Many of these are programming issues. This aspect of the Android problem was highlighted recently with a report from six German academics. 'Why Eve and Mallory love Android: an analysis of Android SSL (in) security' detailed the potential for Man in the Middle (MitM) attacks on Android apps due to the poor implementation of SSL by many app developers.[3] The group's analysis of 13,500 apps found 790 that used SSL – presumably in an effort to provide secure communication – but which would accept any certificate, including self-signed ones. Another 284 apps did at least insist on the certificate being signed by a recognised Certificate Authority (CA) but failed to check whether the certificate had been issued to the domain being contacted. The researchers reported that 1,074 (or 8%) of the apps they examined were vulnerable to a MitM attack.

Much of this, the researchers concluded, boiled down to poor understanding of the protocols, which may be a consequence of the low barrier to entry of the Android platform. This is an aspect that appeals to many: you don't need much in the way of resources in order to develop and market an Android app. By the same token, you also don't need much in the way of experience or knowledge. And understanding of security issues, processes and protocols is often poor even among professional programmers.

Another way in which this problem manifests itself is the excessive permissions that many apps demand. Juniper Networks analysed 1.7 million apps and found that the problem was especially marked with free apps.[4] For example, just over 24% of free apps request permissions that allow the software to track the user's location, while this is true of only 6% of paid apps. Similar discrepancies apply to access to address books, permission to initiate background calls, and the ability to access the camera. Juniper concludes that users need to accept greater exposure of personal data with free apps, although it's difficult to say how many are using excessive permissions for malicious reasons and how many do it because the developers simply add unnecessary capabilities, due to poor awareness of best programming practice.

"Much of this suspicious activity may be nothing more than requesting excessive permissions"

Similarly, Bit9 recently analysed 400,000 apps available via Google Play and characterised a quarter of them as "suspicious" or "questionable".[5] Again, much of this suspicious activity may be nothing more than requesting excessive permissions: Bit9, for example, found that 42% of apps request access to GPS location data.

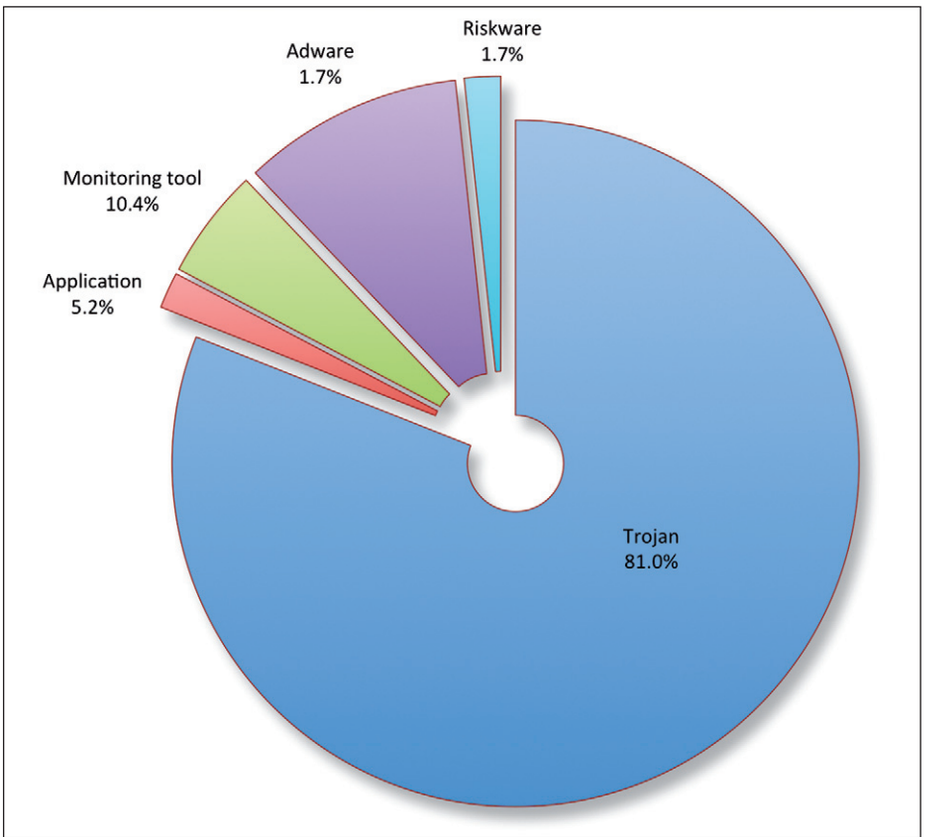

Figure 2: Mobile threats by type, Q2 2012. Source: F-Secure.

Malicious apps

Any platform can suffer from poorly written software. What is of greater concern is the volume of overtly malicious software appearing on Android. The most common way malicious code finds its way on to a device is via a specially crafted app. They may be 'trojanised' versions of legitimate apps that have had malware inserted into them. Or they may be apps written from the ground up as malicious code, although these are still often distributed as fake versions of well-known, legitimate apps. Famous brands exploited in this way have included Instagram, Angry Birds Space, Farm Frenzy, Skype and even (perhaps with some irony) Flash Player. There appears to be a developer community out there supporting the writing of malicious apps: Trend Micro discovered that many of these apps use a common library – *libvadgo* – that provides communications with a Command and Control (C&C) server.[6]

It's very common for malicious apps to pose as networking or communications utilities – purporting to provide wifi, SMS, email and other net-based functions – so that granting the app Internet permissions will appear reasonable. Webroot discovered one – Android Security Suite Premium – that even claimed to be an anti-malware package. While this seemed to presage the arrival of fake-AV on Android, in fact it was just another trojan app: real fake-AV simulates anti-malware functions, such as scanning, and reports the device infected, enticing victims into buying worthless licences. Kaspersky Lab said it believed this trojan app might be a new version of ZitMo – ie, the mobile component of the Zeus banking malware – because of its ability to intercept SMS messages, often used for transaction authentication, and upload them to a remote server.

Pornography is a common enticement to get victims to download apps they might otherwise think twice about. In June 2012, authorities in Japan arrested six men who were allegedly behind a trojanised app that stole personal information while also demanding a fee. The police said that 9,252 people had downloaded the app and 211 of them paid a total of over $250,000 to the cyber-criminals.

Again, none of this is unique to the Google OS: security firm Arxan, which sells tools for hardening applications, insists that there are hacked versions of all the top 100 apps on Android and 92% on iOS. The key difference is that the compromised versions of iOS apps are available only through third-party app stores to users who have jailbroken their devices (and who, therefore, have arguably accepted the risk). Maliciously crafted Android apps, on the other hand, have frequently found their way on to Google Play. And, in any case, downloading from third-party app stores is normal, and often preferred, in the Android ecosystem.

Malicious activity

The malicious activity carried out by the malware is as varied as that on desktop platforms. Harvesting personal information is a common exploit: only recently it was claimed that developers in Japan had cracked popular games apps, inserting data stealing malware, added the words 'The Movie' to the titles and resold them. Early reports suggest that some 90,000 people fell victim to this scam before the developers were arrested. The personal data misappropriated by this kind of software is uploaded to remote servers and might include contact databases, the contents of emails and SMS messages and information about the device itself, such as the IMEI, phone number and so on. This is all useful for identity theft and spamming.

Other forms of malicious activity include the secret dialling of premium-rate phone numbers and similar activity with SMS text messaging. The cyber-criminals take a profit from the premium-rate services while victims end up with huge phone bills. In July 2012, trend Micro said it had found malware that was downloading apps and paid-for media files, incurring charges for the user.[7] And this wasn't the first time malicious apps had been found signing up victims for paid-for services. Spyware is also common, broadcasting the device's GPS location and leaking its call log.

'Spamvertising' has become increasingly common, too. Often this rides the thin line between nuisance

and genuine malware. These apps push advertisements very aggressively. And, of course, many of the ads lead to scams, such as fake anti-virus. In some cases, the ads are disguised as urgent notifications.

"Malware dubbed SMSZombie infected an estimated 500,000 users in China, having been mostly spread through malicious wallpaper files"

The difficulty of determining the fuzzy line that marks the boundary beyond which you're into the realm of criminality was highlighted recently by Sophos, which markets an Android anti-malware package. According to the firm, the most common 'malware' it sees, representing over 63% of the top-five detections, is what it has designated as Andr/PJApps-C.[8] In fact, this isn't malware *per se*: apps detected in this way have been cracked using a widely available tool. Typically, they are genuine apps that have been modified in some way – perhaps to add a malicious payload, but that's not necessarily the case.

Multiple functions

Malware apps often have multiple functions, including some very ominous functions. A recent example was a trojan identified by TrustGo as MMarketPay.A. This was hidden in seemingly legitimate travel and weather apps as well as fake versions of apps from well-known vendors such as Sina and Funinhand, and distributed via third-party app stores in China. Infected devices downloaded paid-for content and apps, but the trojan was also capable of intercepting out-of-band verification SMS messages from China Mobile that are used to enhance security for people making online purchases. The trojan even forwarded Captcha images to a remote server, where they could be analysed by humans.

It's not just apps: malware dubbed SMSZombie infected an estimated 500,000 users in China, having been mostly spread through malicious wallpaper files – many using salacious

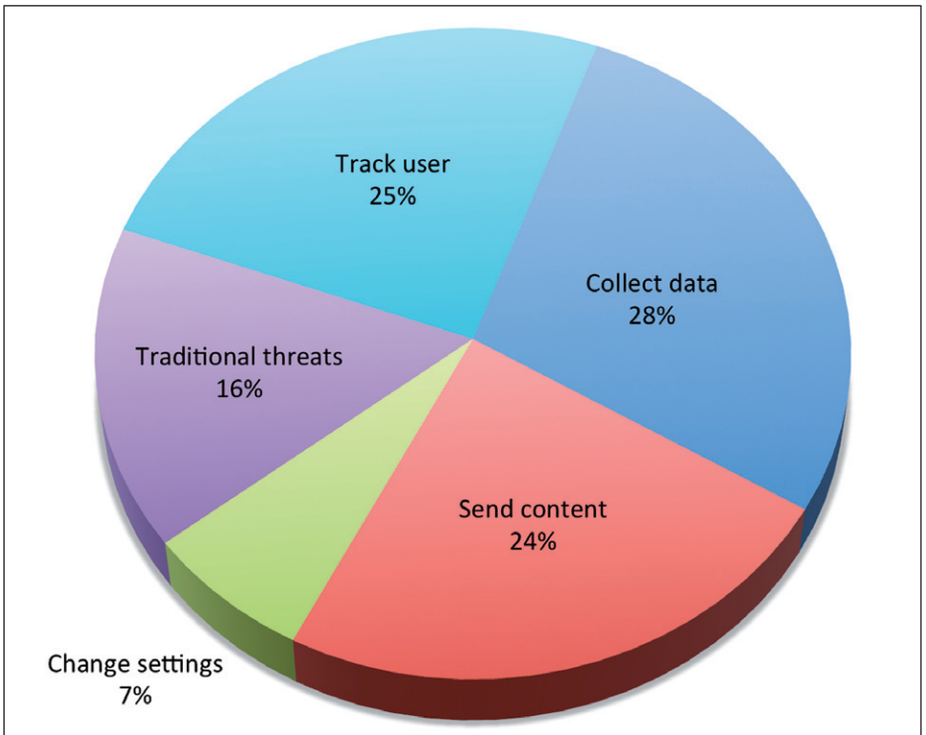

Figure 3: What mobile malware does: figures for 2011. Source: 'Internet Security Threat Report: 2011 Trends', Symantec, April 2012.

images to entice victims. The files are, in fact, apps masquerading as wallpaper and the malware has proven difficult to detect and remove, requiring skills well beyond those of the average Android user, according to TrustGo, which discovered the malware. SMSZombie targeted users of a mobile payment system operated by China Mobile, so the malware hasn't been seen much outside that country, but the technique of intercepting SMS messages used in online transactions could easily be applied to banking and e-commerce systems worldwide.

In May 2012, F-Secure said it witnessed the arrival of the first drive-by infection methods for Android with the NotCompatible.A malware.[9] Just visiting a malicious web page is enough to compromise a vulnerable device. Infection isn't automatic, however: the user finds a notification message and must then agree to install the app. That's where social engineering comes in: the installation application is called com.Security.Update and the app filename is Update.apk. And Chinese mobile security specialist NQ Mobile found a trojan, VDloader, capable of downloading entire apps on to an infected platform.[10] It's delivered as an SMS message. Clicking on the details of the message downloads VDloader – in effect, a dropper that subsequently downloads other apps – malicious or otherwise.

Scale of the problem

It's hard to get reliable numbers with which to judge the scale of infections. Many numbers are thrown about by security vendors. None is provided by Google. But to give an example, the MMarketPay.A found by TrustGo was said to have infected 100,000 devices.

Even worse, the Sexypic app, which was trojanised with the DroidDream malware, has been estimated to have been installed by over 500,000 Android users. Regardless of how accurate (or not) that figure may be, it's still an indication of a serious problem. The app promised to provide pornographic images and therefore requested that the user grant it Internet permissions, which is reasonable. But it also asked permission to send SMS messages, and there is no possible reason why it should need to do that. And yet around half a million people gave it that permission.

Mobile botnets

At the time of writing, there's still considerable debate over whether Android-

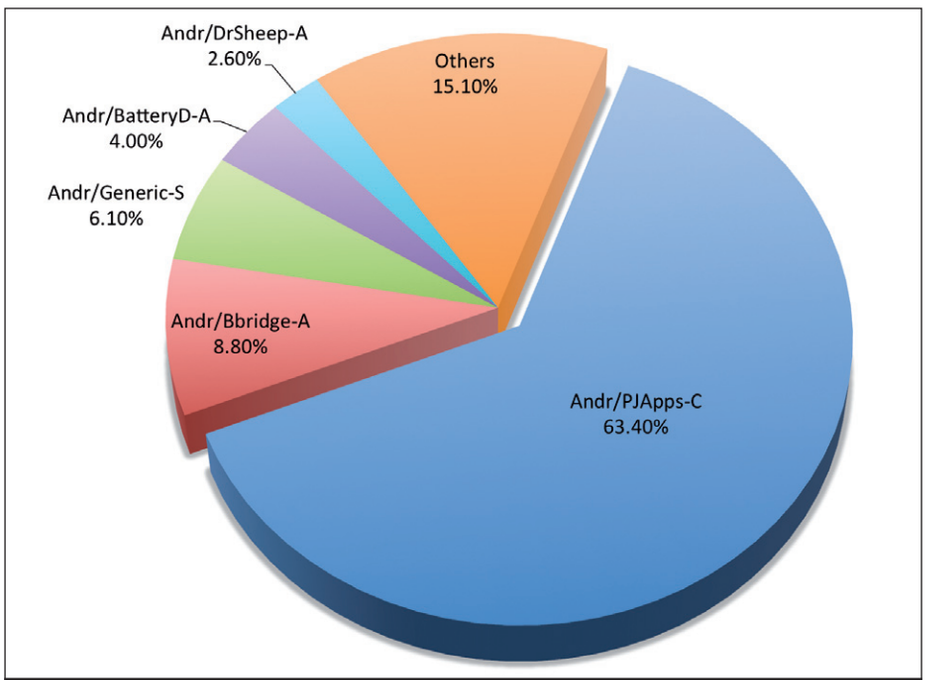

Figure 4: The most commonly detected 'malware', according to Sophos.

based botnets are operating, but the technology is clearly there. McAfee Labs found that a game app – Madden NFL 12 – actually carried a malware dropper capable of installing, among other things, IRC-controlled bot code.[11] F-Secure also announced that NotCompatible.A and the slightly more recent Cawitt.A are capable of turning compromised devices into botnet zombies. Cawitt.A makes a connection to Twitter – which is used as a C&C channel – both for instructions (such as sending SMS messages to premium-rate numbers) and to upload data from the infected device, such as its International Mobile Equipment Identity (IMEI) number.

In July 2012, a heated discussion broke out over the possibility of an Android-based botnet when a number of spam emails, pushing fake pharmaceuticals, were found to contain the string 'Sent from Yahoo! Mail on Android' in the message signature. Some commentators were quick to assume that a botnet running on Android devices was responsible. The first to go public with this suggestion was Terry Zink of Microsoft with a post on his Cyber Security Blog.[12] Google was just as quick to point out that it's easy to insert this string into spam messages sent from PCs.[13] Chester Wisniewski of Sophos, who fanned the flames with a blog post the day after Zink's, pointed out that the spam messages had valid Yahoo Message-IDs and DKIM signatures. A middle ground was suggested by Wisniewski and Kevin Mahaffrey, CTO of Lookout Security, who said his firm found a flaw in Yahoo's email app for Android. Rather than infecting Android devices with bot code, the spammers may have been able to compromise the servers used by the Yahoo! Mail app and were exploiting the company's APIs.

Proof of concept

As with other areas of security, the number of infection vectors and exploits known to and demonstrated by the security community far outstrips those in use in the wild by malicious actors. In a previous article in this series, we discussed the research carried out by the Mobile Exploit Intelligence Project (MEIP). In a presentation at Black Hat Europe, in early 2012, the project's Dan Guido and Mike Arpaia concluded that when it comes to malicious software requiring privilege escalation, the cyber-criminals had come up with no jailbreaks of their own and were dependent on those created by security researchers and the Android jailbreaking community.[14]

Nonetheless, proof-of-concept attacks can help us understand the weak points that could be exploited in the future. For example, a researcher at the University of Luxembourg, Ralf-Philipp Weimann, used the Black Hat USA conference in July 2012 to demonstrate a method of tracking Android users through their devices.[15]

Most smartphones (including iOS) use Assisted GPS (A-GPS) to determine location. In addition to normal GPS signals, they also use location of mobile phone network transmitters (and even known wifi access points) to make a fast location fix. Weimann found that Android phones were using unsecured messages to communicate with mobile phone networks via the Internet. If an attacker shared a wifi-based network with a target phone, it was possible to intercept position fix requests and not only feed back false information before the mobile network responds, but also tell the phone to use a different server in the future. By hijacking the communications in this way, the attacker's server can then trace all future movements of the phone – and its owner.

Xuxian Jiang at North Carolina State University has done extensive work on Android security and recently led a team that demonstrated a clickjacking method that works on all versions of Android up to and including 4.0.4 Ice Cream Sandwich (ICS).[16] Instead of attacking the kernel, it targets the Android framework. Once installed – perhaps by getting the victim to install a fake app – the rootkit intercepts clicks (or taps) and re-routes them. A user tapping, say, the normal and legitimate browser or email icon would instead launch an app of the attacker's choice. This could be, for instance, a specially crafted version of the browser designed to redirect victims to malicious sites or intercept keystrokes, such as login credentials.

Remote wiping

One flaw that got a lot of people's attention presented the potential to remotely wipe phones. A flaw in the handling of Unstructured Supplementary Service Data (USSD) control codes – at first thought to concern only certain Samsung handsets but later found to affect a much broader range of phones

and tablets – could lead to remote resetting of the device. Such codes could be sent from malicious web pages or QR codes. The problem is the automatic execution of USSD codes in versions of Android prior to 4.1.1 Jelly Bean. These codes are normally used by telcos to provide special services, but they could be exploited to lock up devices or deactivate the SIM card, which would need to be replaced. The severity depends on how the specific handset handles the codes: on some Samsung devices, for example, they can cause a full factory reset, wiping all data. Similar codes can be sent to iOS devices, but they don't appear to be passed to the system.

Distribution issues

While the majority of malicious apps are found on third-party app stores – some of which have been established for the sole purpose of distributing malware – many have found their way into Google's own Play portal (previously known as Android Market). When alerted to rogue apps, Google has taken them down, but not always as fast as many people would like. In early 2011, Google pulled more than 50 apps infected with DroidDream but chose not to trigger automatic uninstalls from users' devices.

In February and May 2012, Trend Micro identified 17 malicious apps, 10 using AirPush technology to deliver obtrusive ads to victims, many of them promoting other dubious apps or equally insalubrious websites, and six apps loaded with the Plnkton malware code. Together, these apps were downloaded an estimated 700,000 times before Google reacted and took them offline.

Developer accounts

The large number of malicious apps appearing for Android, and the conspicuous lack of them for iOS, is primarily due to the fact that the iOS platform requires code signing by Apple. When apps have been reviewed by Apple – which includes static review of the code for things like banned APIs and known exploits, and perhaps even testing by humans – the app is signed. A device running iOS will refuse to run any code that doesn't carry Apple's signature or where the signature doesn't match the code. It's not possible to change code after it has been signed without going through the approval and signing process again. Android doesn't use code signing. Although there is app vetting, using Bouncer (of which more later), it's possible for apps that have been installed on users' devices to modify themselves.

However, these are not the only considerations. There are procedural issues to do with how developer accounts are run. With Apple's iOS developer programme, the company will insist on verifying your identity – for example, by demanding that you supply details of your passport or driving licence or, if you're a company, your articles of incorporation. While a malicious app write might be able to fool this system with a fake identity, that's a risky and potentially expensive process, especially when malware apps tend to have short lives and you, as a malware producer, need to repeat this process frequently.

"The popularity of app stores depends on location: in the West, fewer than 10% of Android users will download from third-party stores. In China and other parts of Asia they predominate"

When researcher Charlie Miller managed to sneak a malicious proof-of-concept app into Apple's App Store for iOS, he did it with his own developer account (which was later suspended). When Miller and Oberheide uploaded to Google Play, when they were testing the weaknesses of Bouncer, they used a fake developer account. Play doesn't require any proof of identity.

Third-party app stores

Play and the iOS App Store are not the only games in town. There are numerous app stores for Android. Their popularity depends on location: in the West, fewer than 10% of Android users will download from third-party stores. In China and other parts of Asia, however, they predominate, not least because they are tailored to local languages.

There was some talk, at the beginning of 2012, of developers setting up their own app store for software that Google wouldn't accept – particularly apps aimed at rooted devices. Koushik Dutta, one of the developers behind the CyanogenMod firmware for Android phones, mooted the idea but so far it has come to nothing.

Apple iOS users have third-party stores, too, the most popular being Cydia. However, these can be used only for jailbroken devices. Ironically, this makes such stores very attractive to malware writers: they know that any app downloaded from such a store is going on to an already jailbroken device.

At time of its presentation at Black Hat Europe 2012, MEIP had seen no genuine malware on iOS from the Apple App Store but 30 families of malware in Google Play. And it had seen 20 malware campaigns being conducted through US third-party app stores and 32 via their Chinese equivalents.

Evading Bouncer

In February 2012, Google took steps to improve security in its (then) Android Market online store by introducing Bouncer.[17] This is an automated tool that checks apps submitted to the app store for suspicious activity. Google's own blog post, by Hiroshi Lockheimer, VP of engineering, Android, acknowledged the problem in an oblique way. Lockheimer said that Bouncer had been in action, unannounced "for a while now" and that, "between the first and second halves of 2011, we saw a 40% decrease in the number of potentially malicious downloads from Android Market". Alas, Bouncer has proven to be relatively easy to avoid and it wasn't long until researchers produced proof-of-concept evasion techniques.

The first was Charlie Miller, this time teamed with Jon Oberheide of Duo Security. At SummerCon 2012 they announced that they'd discovered "at least 20" ways of evading or subverting Bouncer. They were caught by Google a couple of times during their research,

but only when they were being particularly flagrant, they said.

Potential techniques for evading Bouncer could be based on attacking the technology directly, the pair suggested – for example, by attacking known vulnerabilities in QEMU, the emulator used by Bouncer to run a virtual Android environment. Or malicious apps could detect that they're being run in the virtual environment – perhaps by something as simple as checking for the existence of the /sys/qemu_trace directory – and simply remain quiet. This could be very effective because Bouncer appears to rely on dynamic analysis, detecting malicious behaviour rather than the static analysis approach of looking for suspicious code.

The researchers were able to fingerprint Bouncer by uploading an app to Play that opened a remote shell, allowing them to explore the environment in which the app was running while being examined by Bouncer. Among other things, they discovered that (at that time – presumably Google has since changed this), Bouncer simulated a device with a Google account of Miles.Karlson@gmail.com, with only one contact in its address book and just two stored images – one of Lady Gaga and another of a cat. Bouncer would watch to see if the app attempted to exfiltrate these files.

One evasion technique is very simple: at the time the researchers were carrying out their investigations, they found that Bouncer ran each app for only five minutes. If an app delayed carrying out any suspicious activity for that length of time, it would pass inspection.

JavaScript trick

Nicholas Percoco, head of Trustwave SpiderLabs, and Sean Schulte, an SSL developer at Trustwave, also managed to sneak malware past Bouncer. In a presentation at Black Hat USA in Las Vegas, July 2012, they demonstrated a JavaScript trick capable of turning an apparently innocent app into a malicious one after it had been accepted into Google Play.

The pair created a fake SMS blocker app – one that would, ostensibly, allow users to prevent people sending them unwanted SMS text messages. In fact, 'SMS Bloxer' was effectively a malware dropper. To discourage innocent Android users from buying and installing the app, the researchers priced it at $49.95: this class of app more typically sells for $3 or less and there are free versions.

"Bouncer did eventually issue an alert, but only after the researchers had pushed the app to performing malicious activity every second"

The initial version of the app contained no malicious content and was therefore approved by Bouncer without issues. SMS Bloxer was in the Play app store for two weeks before the researchers removed it. Over that time, the app used the JavaScript bridge technique approved by Google for updating apps, including both adding new features and modifying HTML code to change the appearance. Such changes did not require re-approval. In this way, the app added new – and malicious – functions, such as the ability to steal contact data, force the loading of a given web page, and send SMS messages.

"Some people may be as dubious about the efficacy of anti-malware on mobile platforms as they are about its poor record on desktop machines"

Apps appear to be periodically re-checked by Bouncer, so the researchers took the precaution of finding Bouncer's IP address. The app would then perform suspicious activity only when run on a device with a different address. Bouncer did eventually issue an alert, at which point Google suspended the developer account, but only after the researchers had pushed the app to performing malicious activity – uploading the user's address book to a remote server – every second. The researchers shared their findings with Google and reported a positive response.

Countering malware

It's not all doom and gloom. There are positive steps users can take and signs that the platform itself is starting to deal with the issues. For a start, all the major anti-malware software vendors offer solutions for Android. Uptake appears to be currently very low – Trend Micro has estimated it at 20% – and for a number of reasons. These include a lack of awareness of the need for or availability of anti-malware software on this platform, and perhaps a dislike for the performance hit that such tools can cause. And some people may be as dubious about the efficacy of anti-malware on mobile platforms as they are about its poor record on desktop machines. Nevertheless, at the time of writing there was a suggestion that Google itself might be adopting anti-malware scanning – not in the operating system, but on Play. This may, or may not, be linked to its acquisition of VirusTotal.

With Android 4.1 'Jelly Bean', Google introduced app encryption as a copy protection technology.[18] Paid-for apps are encrypted using a key unique to each device, but before the app is downloaded. This prevents the .apk file being run on other devices. The app is also stored in an encrypted directory, unlike normal apps which are normally stored in the /data/app partition. However, it does nothing to prove the validity of the app, which is not signed. Google also hit a problem with a number of apps failing to run from the encrypted directory and had to temporarily deactivate the copy protection system while developers amended their apps to run from the new location.

Jelly Bean also saw the introduction of proper Address Space Layout Randomisation (ASLR). The previous version of the OS, 4.0 Ice Cream Sandwich (ICS), used ASLR for some features, but app code and the

linker were still loaded at fixed and predictable memory locations. As part of this security improvement, version 4.1 also gained Position Independent Executable (PIE) support – a key element of ASLR, and other features, such as the enabling of dmesg_restrict and kptr_restrict to avoid leaking kernel addresses. However, some researchers have questioned the value of ASLR against the kinds of threats seen on Android: Oberheide of Duo Security blogged that attackers would find ways to bypass it, for example by exploiting the weak entropy of the 32-bit address space.[19]

App developers can sign their own code, but cyber-criminals who hack the apps in order to append a malicious payload can get around this fairly easily by disabling the certificate checks, or by adding their own self-signed certificates.

Research tools

The burgeoning interest in Android security is leading to the creation of a variety of tools that will help researchers and developers understand weaknesses in existing software and also develop more robust code in the future. In a previous article, for example, we described the Mercury Framework, which can be used to analyse the attack surface of apps.[20] Another tool – the open source Android Security Evaluation Framework (ASEF) – has been created by Parth Patel, a vulnerability signature engineer at Qualys, to check what Android apps are up to.[21] Among other things, it alerts you to unusual activity or behaviour for which you have not given specific permission.

MEIP's research and Trend Micro's discovery of a library common to much Android malware, suggests that the sources of malicious software for Android might be relatively few. Xuxian at NC State is leading a project, assisted by a number of universities and vendors, to map the Android malware 'genome'.[22] The aim is to establish a framework for sharing Android malware samples among researchers in order to analyse and categorise the various families of malicious code. Membership of and access to the Android Malware Genome Project is available only to vetted users.

"The US National Security Agency (NSA) has released a hardened version of Android – SE Android"

There's also a inherently more secure version of Android – though not from Google. In a reprise of its pivotal role in the creation of the Security-Enhanced Linux (SELinux) project, the US National Security Agency (NSA) has released a hardened version of Android. SE Android uses many of the same mandatory access controls as SELinux to enforce strict access control policies. The resources available to any specific app can be more narrowly defined – system resources that all apps can access by default in regular versions of Android can be denied in the NSA implementation. For example, the volume daemon 'vold', which runs with root privileges, is exploited by GingerBreak to root a device. By cutting off access to such vulnerable components, SE Android can halt such exploits in their tracks, although how effective it is depends entirely on configuration. SE Android is unlikely to have much effect on the Android world in general. Its main users will be government departments and organisations, such as defence contractors, that need to work within federally mandated security strictures.

Conclusion

Android represents a kind of frontier territory in which a general lawlessness has resulted in both a vibrant and innovative community – encompassing developers and users – and significant dangers. To what degree the threats can be countered depends, to some extent, on how much interference and control Android users will tolerate. Certainly, typical Android users and developers could not stomach what they see as the oppressively policed walled garden of Apple's iOS platform, for all its security benefits. And the future of security on Android will also be shaped by the severity of exploits. After all, Windows has suffered badly from malware, but the OS is still in common use.

About the author

Steve Mansfield-Devine is a freelance journalist specialising in information security. He is the editor of Network Security *and its sister publication* Computer Fraud & Security. *He is also a Certified Ethical Hacker.*

References

1. Android Under Siege: popularity comes at a price'. TrendLabs 3Q 2012 Security Roundup. Oct 2012. www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-3q-2012-security-roundup-android-under-siege-popularity-comes-at-a-price.pdf.
2. OWASP Mobile Security Project, Top 10 Mobile Risks. Accessed Oct 2012. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project.
3. Fahl, S; Harbach, M; Muders, T; Smith, M; Baumgärtner, L; Frieslaben, B. 'Why Eve and Mallory love Android: an analysis of Android SSL (in)security'. Accessed Oct 2012. http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf.

4. Hoffman, Daniel. 'Exposing your personal information – there's an app for that'. Juniper Networks blog, 30 Oct 2012. Accessed Nov 2012. http://forums.juniper.net/t5/Security-Mobility-Now/Exposing-Your-Personal-Information-There-s-An-App-for-That/ba-p/166058.
5. 'Pausing Google Play: more than 100,000 Android apps may pose security risks'. Bit9, Nov 2012. Accessed Nov 2012. https://www.bit9.com//files/1/Pausing-Google-Play-October2012.pdf.
6. Sun, Weichao. 'Library file in certain Android apps connects to C&C servers'. Trend Micro blog, 11 Jun 2012. Accessed Aug 2012. http://blog.trendmicro.com/library-file-in-certain-android-apps-connects-to-cc-servers/.
7. Sun, Weichao. 'Android malware family downloads paid media and apps'. TrendLabs Security Intelligence blog, 17 Jul 2012. Accessed Nov 2012. http://blog.trendmicro.com/android-malware-family-downloads-paid-media-and-apps.
8. Cluley, Graham. 'Revealed! The top five Android malware detected in the wild'. Naked Security, Sophos, 14 Jun 2012. Accessed Aug 2012. http://nakedsecurity.sophos.com/2012/06/14/top-five-android-malware/.
9. 'Mobile Threat Report Q2 2012'. F-Secure. Accessed Oct 2012. http://www.f-secure.com/weblog/archives/MobileThreatReport_Q2_2012.pdf.
10. 'Watch out for VDloader – new malware delivers bad apps via SMS'. NQ Mobile blog, 31 Jul 2012. Accessed Aug 2012. http://en.nq.com/blog/?p=1364.
11. Sabapathy, Arun. 'Evolution of Android malware: IRCBot joins the party'. McAfee, 9 May 2012. Accessed Aug 2012. http://blogs.mcafee.com/mcafee-labs/evolution-of-android-malware-ircbot-for-android.
12. Zink, Terry. 'Spam from an Android botnet'. Terry Zink's Cyber Security Blog, 3 Jul 2012. Accessed Aug 2012. http://blogs.msdn.com/b/tzink/archive/2012/07/03/spam-from-an-android-botnet.aspx.
13. Wisniewski, Chester. 'Android spam bots? What we know for sure'. Sophos Naked Security blog, 6 Jul 2012. Accessed Nov 2012. http://nakedsecurity.sophos.com/2012/07/06/android-spam-bots-what-we-know-for-sure/.
14. Mansfield-Devine, Steve. 'Paranoid Android: just how secure is the most popular mobile platform?'. Network Security, Volume 2012, Issue 9, Sept 2012, pp.5-10. Accessed Oct 2012. www.sciencedirect.com/science/article/pii/S1353485812700818.
15. Simonite, Tom. 'GPS weakness could enable mass smartphone hacking'. Technology Review, 26 July 2012. Accessed Aug 2012. http://www.technologyreview.com/news/428632/gps-weakness-could-enable-mass-smartphone-hacking/.
16. Shipman, Matt. 'Clickjacking rootkits for Android: the next big threat?'. The Abstract, NC State University, 2 July 2012. Accessed Aug 2012. http://web.ncsu.edu/abstract/technology/wms-jiang-clickjack/.
17. Lockheimer, Hiroshi. 'Android and security'. Google Mobile Blog, 2 Feb 2012. Accessed Aug 2012. http://googlemobile.blogspot.fr/2012/02/android-and-security.html.
18. 'Android 4.1 for Developers'. Android.com. Accessed Nov 2012. http://developer.android.com/about/versions/jelly-bean.html.
19. Oberheide, Jon. 'Exploit mitigations in Android Jelly Bean 4.1'. The Duo Bulletin, 16 July 2012. Accessed Aug 2012. https://blog.duosecurity.com/2012/07/exploit-mitigations-in-android-jelly-bean-4-1/.
20. Mansfield-Devine, Steve. 'Android architecture: attacking the weak points'. Network Security, Volume 2012, Issue 10, Oct 2012, pp.5-12. Accessed Oct 2012. www.sciencedirect.com/science/article/pii/S1353485812700922.
21. Android Security Evaluation Framework. Accessed Nov 2012. http://code.google.com/p/asef.
22. Shipman, Matt. 'Announcing the Android Malware Genome Project'. The Abstract, NC State, 22 May 2012. Accessed Aug 2012. http://web.ncsu.edu/abstract/technology/wms-android-genome/.

EVENTS CALENDAR

26 November–3 December 2012
SANS London 2012
London, UK
www.sans.org/event/london-2012

3–5 December 2012
ASIS 6th Asia-Pacific Security Forum and Exhibition
Hong Kong
http://bit.ly/PvyG48

10–13 December 2012
Black Hat Abu Dhabi 2012
Abu Dhabi, UAE
www.blackhat.com

15 December 2012
Security BSides Seattle
Seattle, Washington, US
http://bit.ly/Rve5Rl

16 January 2013
Securi-Tay 2
Abertay Dundee University, UK
http://securi-tay.co.uk/

28–31 January 2013
Cyber Defence and Network Security 2013
London, UK
www.cdans.org

11–13 February 2013
3rd Annual Business Continuity & Emergency Response Forum
Abu Dhabi, UAE
www.fleminggulf.com

19–22 February 2013
OWASP AppSec AsiaPac 2013
Jeju, South Korea
http://bit.ly/SVp1cx

24–25 February 2013
Security BSides San Francisco
San Francisco, US
www.securitybsides.com/w/page/35868077/BSidesSanFrancisco