6.857 Final Project Proposal: Mobile Device Security

Liz Fong, David Lam, David Wang, & Mark Yen
{lizfong, d_lam201, wang4, markyen}@mit.edu

March 17, 2011

1 Introduction

Mobile devices such as smartphones are becoming increasingly popular, and are set to outstrip sales of PCs for the first time this year. This means that they are also becoming bigger targets for hackers and criminals. One study found that from May to December 2009, malware and spyware on mobile phones doubled from 4% to 9%. Still, consumers are not treating smartphones with the same care as they do with their PCs (few users bother to install anti-virus software on their phones, for instance, and the effectiveness of anti-virus software for phones is questionable at best).[1]

However, the number of vulnerabilities in mobile devices is no less than their desktop counterparts. A test conducted by viaForensics found that 10 out of 12 e-mail applications tested failed to store usernames and other information in a secure way.[2] Also, when communicating over the network, a security professor at Rice University found that Android apps like Facebook and Google Calendar send their information in the clear over the network, even if the accounts are set to always use SSL on the desktop versions.[3]

Finally, loopholes have been discovered in the Android Market which allow seemingly innocent applications to automatically phone home and download malicious payloads.[4]

[1] Richmond, Riva. "Security to Ward Off Crime on Phones", *The New York Times*, February 23, 2011. http://nyti.ms/f4LQDi

[2] Kolesnikov-Jessop, Sonia. "Hackers Go After the Smartphone", *The New York Times*, February 13, 2011. http://nyti.ms/e0uVmi

[3] Goodin, Dan. "Security shocker: Android apps send private data in clear", *The Register*, February 24, 2011. http://www.theregister.co.uk/2011/02/24/android_phone_privacy_shocker/

[4] Greenberg, Andy. "Researcher Builds Mock Botnet Of 'Twilight'-Loving An-

Even after the loopholes had been patched, it appears that there is little to no screening done on the applications uploaded to the store, and on March 5, Google saw 58 malicious apps uploaded to the Android Market. These apps were corrupted versions of legitimate products, and so they carried names such as Super Guitar Solo, Advanced Barcode Scanner, and Bubble Shoot. Downloaded to around 260,000 devices before Google removed them from the Market, the applications contained malicious code that could compromise personal data such as the IMSI number of the phone.[5] This incident led Google to exercise its previously unused ability to remotely remove applications from its users phones, in order to protect them from the dangerous applications. Google has stated it will be making changes to prevent similar malicious applications from being distributed through those markets, though it did not go into detail on what those changes were.[6]

Another problem is the lack of quality-control in third-party app stores. Google has not officially launched their Android Market in China, so a number of third-party app stores have popped up to fill the void. However, an analysis of those markets found only 61% of those apps to be unique, 36% to be redistributed, and 2% to be pirated.[7] A number of those apps distributed on these third-party app stores are in fact apps repackaged as trojans, which can then hijack certain functions of your phone.[8]

The trend is set to continue. Ed Amoroso, chief security officer at AT&T, has remarked that 2011 is the "eye of the storm", as 4G network speeds start to make hacking more and more attractive to criminals.[9] According to a report from ICSA Labs, "while most hackers heavily focused on Nokia's mobile phones [in 2010], mobile malware will increasingly target non-Nokia devices including Apple, Blackberry, Android, and Microsoft." And according to Adam Powers, CTO of Lancope, "perimeter-based defenses, such as firewalls and IPS, aren't enough anymore. Corporations must think about how they will deal with smart phones, WiFi devices, and other consumer-oriented mobile devices."[10]

droid Users", *Forbes*, June 21, 2010. `http://blogs.forbes.com/firewall/2010/06/21/researcher-builds-mock-botnet-of-twilight-loving-android-users/`

[5]Ante, Spencer E. and Efrati, Amir. "Google Takes Heat Over App Security", *Wall Street Journal*, March 8, 2011. `http://on.wsj.com/ga73Tn`

[6]Menn, Joseph. "Google disables Android malware", *Financial Times*, March 7, 2011. `http://www.ft.com/cms/s/0/f04a88b8-48ea-11e0-af8c-00144feab49a.html`

[7]Lookout Mobile Security. "App Genome Report", February 2011. `http://bit.ly/idg1tI`

[8]Rothman, Wilson. "Smart phone malware: The six worst offenders", *MSNBC Technolog*, February 16, 2011. `http://technolog.msnbc.msn.com/_news/2011/02/16/6063185-smart-phone-malware-the-six-worst-offenders`

[9]Messmer, Ellen. "Do wireless providers like Verizon and AT&T crimp mobile security?", *Network World*, February 18, 2011. `http://www.networkworld.com/news/2011/021811-verizon-att-mobile-security.html`

[10]Wilson, Tim. "For Hackers, 2011 Looks Like a Prosperous New Year", *Darkreading*, January 3, 2011. `http://www.darkreading.com/security/vulnerabilities/228901590/for-hackers-2011-looks-like-a-prosperous-new-year.html`

For our final project, we have five areas that we plan to investigate:

1. **Network information leakage**: data and personal info that can be gained by a network eavesdropper without direct access to the device
2. **App-to-app security**: protection of sensitive application data from hostile apps residing on the same device
3. **Granular userspace permissions**: extending the Android system permissions framework to allow applications to request very specific accesses, and implementing a security daemon in userspace to allow even older versions of the Android platform to take advantage of more granular permissions.
4. **Stolen smartphones and lock codes**: examining how resistant to attack the data on smartphones is once it is physically in the hands of an adversary, even with lock codes. Evaluation of alternate screen lock mechanisms accessible to persons with disabilities.
5. **App marketplace security**: determine what types of malware can pass Google's and Apple's checks and become available for download at the official marketplace

2 Network information leakage

For this task, we intend to build on the findings that Dan Wallach (professor of computer security at Rice University) posted on his blog in February.[11] Using Wireshark and Mallory, Wallach tested Facebook, Twitter, several Google apps, and a couple other applications. He found that while Gmail and Google Voice encrypt your traffic, Google Calendar, Google Reader, Google Maps, Google Goggles, Twitter, and Facebook send all of their data in the clear, and the data you are currently working with can be intercepted by others monitoring the network traffic.

We plan to start with Facebook, and break down exactly what calls are being made when. From preliminary data gathering, we have determined that several of the calls to the Facebook server are using undocumented parts of the API. We would like to see if we can reconstruct the entire API and make requests against it ourselves. We would also like to see

[11] Wallach, Dan. "Things overheard on the WiFi from my Android smartphone, February 22, 2011. `http://shar.es/3UVsP`

if we can spoof notifications to the mobile device, for instance, chats, messages, or wall posts that appear to come from arbitrary Facebook users. We would also like to compare the differences between the iOS version of the Facebook application and the Android version of the application. We know from preliminary testing that they use different versions of the API and in some cases different protocols.

The next application we would like to test would be Google Maps, which communicates not only the user's current fine GPS location every few seconds, but also all of the user's friends' locations on Google Latitude. It would be very interesting to see exactly how much and under what conditions this location data is sniffable.

3 App-to-app security

Here we try to build on the testing done by viaForensic's appWatchdog project.[12] They have found several applications that store sensitive data insecurely. These applications include: Mint, Groupon, Kik, Android Mail, and iPhone Mail.

We would first like to install these applications ourselves and verify what data is actually accessible with physical access to the phone, since appWatchdog is short of details. Secondly, we would like to attempt to construct a malicious application that would try to read the sensitive data we find without any intervention from the user. If this is possible, it would also be interesting to see what permissions the user is required to approve before it we can pull this off.

Again, we would like to do testing on both iOS and Android, so that we can get a sense of their robustness in security in relation to one another.

4 Granular userspace permissions

The permissions scheme in the Android market results in substantial confusion to users and forces application developers to request far more permissions then their applications need to run. Many users will simply click through the permissions request screen upon application installation in order to more quickly use the application they've chosen, no matter how scary the permissions being requested are. Applications that require access to specific ad servers and rough geographic location to serve their in-application advertisements instead

[12] viaForensics. appWatchdog, February 2, 2011. `http://viaforensics.com/appwatchdog/`

are forced to request access to the entire internet without limitations and coarse GPS access. Although more granular permissions could be coming to Gingerbread and Honeycomb, over 98.1% are still running applications on Donut, Eclair, and Froyo[13]. Thus, the only possibility for implementing granular permissions is in userspace rather than as a core component of the platform's security framework.

We propose to create a userspace service that has full permissions and delegates those permissions to client applications after consulting user-specified preferences and blacklists to evaluate the risk of allowing a given permission to a specific resource. We will create sample applications, both non-malicious and malicious, and demonstrate that the malicious applications cannot not hijack the service, and that the non-malicious applications can request a reduced subset of permissions and still function.

The client application will send messages to the userspace security service to request permissions be delegated to it (e.g. "com.google.sites.androidapp needs to open an https socket to https://sites.google.com"). The userspace security service checks its existing records of whether the user has previously authorized the application for that URL. If yes, then the connection is created in the userspace service (using the permissions originally granted by the base OS permissions framework) and is passed to the client application. If not, the user is prompted to choose whether to allow the request once, to always allow https://sites.google.com access for com.google.sites.androidapp, or whether to deny the url request. Additional checks could be implemented such as checking the Google SafeBrowsing API to warn the user if an application is attempting to contact a known domain associated with malware. The user, or parties the user trusts, could maintain lists of known behavior by whitelisted applications to import as sensible defaults in order to limit the amount of user interaction required for initial runs of an application. Other such permissions include access to specific files on the device, access to the user's contact list, etc.

The client application needs no permissions of its own at install time since its only interactions are through the userspace security service, and thus will not need to be authorized by the regular OS permissions framework/market system.

[13]Google, Inc., March 15, 2011. `http://developer.android.com/resources/dashboard/platform-versions.html`

5 Stolen smartphones and lock codes

Over 164,000 mobile phones are stolen, lost, or damaged each day in the United States[14], and approximately 5,000 of those are stolen smartphones [15]. We are interested in exploring what a determined adversary can accomplish with physical control over a smartphone. In particular, we would like to see if screen lock and remote wipe functionality can be bypassed by immediately pulling the battery/sim from a smartphone before the owner can report a missing device, and if the screen lock can be bypassed with booting off the SD card slot. We will explore what data is stored unencrypted on the SD card and on the phone's USB-accessible flash. Lastly, we would like to see if a less intrusive but equally secure lock/unlock mechanism can be devised that will enable "busy" users who don't lock their screens due to convenience or users with disabilities that cannot use standard passcode or swipe security for physical reasons to lock their phones against casual attack.

6 App marketplace security

Finally, if we have extra time after completing everything described above, the last goal of our project is to test the approval process for Apple's App Store and for Google's Android Market. We can submit a series of applications that attempt to take more and more questionable actions and see which ones make it onto the store.

We would like to explore findings of security researcher Jon Oberheide of Duo Security, who added an application disguised as a Twilight Movie Preview app to the Android Market.[16] The application contained hidden instructions to "phone home" to a server under Oberheide's control to check for new payloads, download them to the user's device, and execute them. This means any time a new vulnerability in the kernel is discovered, a payload exploiting it could be downloaded and the phone rooted. This assumes of course that the malicious party releases exploits quicker than the network carrier releases patches, but given recent track record of the carriers' OTA updates, this is not difficult at all for the malicious party. According to Oberheide, "[i]t's absolutely trivial to win this race."

In response to the first outbreak of 58 real trojans on their Market in early March 2011, Google announced their had taken steps towards preventing a similar exploit from happen-

[14] Asurion, `http://www.gottahavemymobile.com/it-could-happen-to-you/`

[15] Hartland, Hayden, April 29, 2010 `http://www.emrandhipaa.com/emr-and-hipaa/2010/04/29/guest-post-will-your-new-smartphone-ruin-your-practice/`

[16] Oberheide, Jon. "Remote Kill and Install on Google Android", June 25, 2010. `http://jon.oberheide.org/blog/2010/06/25/remote-kill-and-install-on-google-android/`

ing in the future.[17] However, they never provided any details on what those precautions might be, and we would like to get a better sense of what can currently get through and what would get rejected.

Ideally we would test on both the Android Market and on the App Store; however, this may be difficult in practice because to publish to the Android Market and the App Store require $25 and $85 developers fees respectively, which would prevent us from creating multiple accounts to publish from if we were to get blocked. We are working on acquiring permission from Rich Cannings at Google to perform tests in the live Android Market.

[17] Cannings, Rich. "An Update on Android Market Security", Google Mobile Blog, March 5, 2011. `http://bit.ly/dWvNU7`