Smartphone Security: Trends and Predictions

Dan Wallach, Rice University

February 17, 2011

1 Introduction

It's no exaggeration to say that smartphones, whether RIM's Blackberry, Apple's iPhone, Google's Android, or other models coming out every day, have become ubiquitous in government and in the population at large, and it's no wonder. For the worker on the move, everything from email to calendering, as well as news and entertainment, is now available in a convenient pocketable device. These devices allow users to remain productive, even when walking between destinations, stuck on public transit, or during downtime in meetings. Clearly, they're here to stay. This think piece considers the various security risks and opportunities that arise with the ubiquity of modern smartphones.

Trend 1: smartphones are real computers. Current-generation smartphones generally use ARM microprocessors with clock rates over 1GHz, with new models coming soon that have dual or quad-core CPUs and even higher clock rates. They have gigabytes of storage and reasonably fast networks. In these respects, current smartphones are faster and better-provisioned than desktop computers from a decade ago. This raises an interesting opportunity because smartphones have more than adequate resources to leverage decades of research into secure operating systems. Smartphones from most vendors can or will soon run full-blown Unix-style operating system kernels. This means that security features ranging from virtualization and secure booting to information flow control and multi-level security can potentially be applied to creating high-assurance software within our phones.

Trend 2: many government, military, and private sector IT departments are locking personnel out of social network sites, pushing those users to do these activities on personal smartphones. The justification for these actions vary, but a recent order from the Marines is fairly typical:

> Internet SNS are defined as web-based services that allow communities of people to share common interests and/or experiences (existing outside of DoD networks) or for those who want to explore interests and background different from their own. These Internet sites in general are *a proven haven for malicious actors and content* and are particularly high risk due to information exposure, user generated content and targeting by adversaries. The very nature of SNS creates a larger attack and exploitation window, *exposes unnecessary information to adversaries and provides an easy conduit for information leakage* that puts OPSEC, COMSEC, personnel and the MCEN at an elevated risk of compromise. Examples of Internet SNS sites include Facebook, MySpace, and Twitter.

> . . . access is hereby prohibited to Internet SNS from the MCEN NIPRNET, including over virtual private network (VPN) connections.
>
> —Marine Corps order [6], emphasis mine

This Marine Corps order may protect the MCEN internal network, but it fails to protect Marine personnel, who will continue visiting social network sites, albeit now on their smartphones or home computers. For personnel who work in environments where smartphones are banned, they may well go outside to use their phones on breaks. As such, all that these policies accomplish is pushing "dangerous" behavior from internal systems, where administrators may be able to monitor them, to external systems, where attacks may well go unnoticed.

Trend 3: "Safe" web sites can and will be compromised. In environments such as the MCEN NIPRNET, where social networking is banned, a typical firewall policy is to ban sites known to be undesirable and allow everything else (sometimes called a "blacklist" policy). More aggressive IT managers may default to banning everything, by default, and only allowing a handful of known-good sites (sometimes called a "whitelist" policy). The problem with blacklists is that they're always missing something bad. The problem with whitelists is that any arbitrary web site may fall victim to a security-related attack.

To pick a notable example, visitors to the the New York Times's web site were attacked in September 2009 by a malicious advertisement that tried to trick the user into installing fake antivirus software [12].

> The creator of the malicious ads posed as Vonage, the Internet telephone company, and persuaded NYTimes.com to run ads that initially appeared as real ads for Vonage. At some point, possibly late Friday, the campaign switched to displaying the virus warnings.
>
> Because The Times thought the campaign came straight from Vonage, which has advertised on the site before, it allowed the advertiser to use an outside vendor that it had not vetted to actually deliver the ads, Ms. McNulty said. That allowed the switch to take place. "In the future, we will not allow any advertiser to use unfamiliar third-party vendors," she said.
>
> Mr. Frons said it was unclear how many people saw the ads.

The advertisement displayed what appeared to be a virus scanner, really just a graphical simulation of one, running inside the browser. It then claimed to have found viruses on the computer and offered to allow the user to install software to clean up the infection.

> Security experts say that people who followed the ads' instructions and installed the fake antivirus software will likely receive periodic offers to buy more types of software. (Most legitimate antivirus programs are able to clean up the mess left behind.)
>
> "Once they've fooled you with one thing, they try and fool you with something else," said Kevin Haley, a director in the security software maker Symantec's response team. "It's extremely profitable for them."

This attack had no particular target population in mind, but an adversary interested in attacking the Marines or any other branch of the military could mount a similar attack, using similar means, only selecting a web site whose audience is more likely to be military personnel. At the point where an attacker has convinced military personnel to install third-party software on their smartphone, the risks are quite substantial. Modern smartphones have GPS hardware, to measure their precise location. These phones also have microphones

and antennas, making them excellent remotely-operated listening devices. (Commercial software companies, such as FlexiSpy, already offer phone-modification products that do exactly this. Apple even offers a service to allow users to find "lost" phones, which could clearly be used in a variety of other ways.) Smartphones also have within them the necessary user names, passwords, or other credentials to connect to remote mail servers and otherwise allow an attacker to go quite far into attacking an enterprise.

Thesis: The modern smartphone is part of the military and civilian attack surface and needs to be treated as such. The remainder of this paper considers how we might be able to better manage users' needs to visit arbitrary web sites (social networking and otherwise), how we can engineer smartphones to better work with the security infrastructure we already have in place for desktop computers, and what open research topics remain in managing and mitigating against these sorts of threats.

2 Threat models

Before we delve into particular technologies for smartphones, we should consider the threats that their users will face. For the purposes of this document, we will segment users into two distinct populations: low-value targets (LVT) and high-value targets (HVT), with the core difference being that HVTs are sufficiently attractive that adversaries will consider attacks that require physical access to an HVT's phone, acquired temporarily via espionage of one sort or another. LVTs, on the other hand, may still be worth attacking, but not enough that an attacker would go after them in person. Speculation as to what classes of government, military, or corporate personnel may be classified better as LVT vs. HVT is beyond the scope of this document, although it's certainly safe to say that *most* government and military personnel are in the LVT category.

With these LVT and HVT definitions, several things become clear. HVT users must worry about their phones being surreptitiously modified or even being stolen and replaced with replicas. An HVT phone must then resist physical tampering, including logic to "zeroize" itself when an attack is in progress. (This think piece will discuss the SME-PED program in Section 3, which is intended to be robust against HVT threats.)

Of course, a person's LVT vs. HVT status may change as they travel. A visit to an overseas country known to have the capability and motive to attack smartphones can convert an LVT target into an HVT target, if only for the duration of the trip. Luckily, mitigations such conditions can be very simple to implement: leave your laptop and cellular phone at home, perhaps giving out "loaner" gear for the duration of the trip.

The LVT threat model still covers a lot of ground. People will still be subject to broad-spectrum attacks from hackers and the like, mostly interested in their credit card numbers, banking passwords, and other ways they can make money. In addition to such broad attacks, various personnel may well be individually and personally targeted, albeit not in the flesh. A foreign intelligence agency, for example, may well have a strong interest in turning White House staffers' phones into bugging devices, but they may not have the means to physically acquire and tamper with those staffers' phones. Even if they do have the means, they may not wish to risk a physical operation that could endanger their agents. Purely electronic attacks, because they can potentially be conducted with some measure of anonymity and from afar, may be judged to offer less risk to the attacker, and thus would be more attractive to pursue.

It's also important to note that any user might be *tricked* into attacking themselves, which could be as simple as being given a series of instructions for how to install the latest cool game. Also, consider users who connect their phones to their personal computers. An iPhone must be regularly connected to its host computer to synchronize with iTunes. Even for Android phones, which synchronize over the air, users may well just to charge their phones from a computer's USB port. In these cases, a security vulnerability in the

computer could well be exploited, giving the attacker access to whatever control is available via the USB port. For many phones, this is sufficient to mount an HVT attack.

The key distinction of the LVT threat model is that it allows us to build a "trusted computing base" (TCB) into the phone. If we trust the phone hardware, then we can build a secure operating system kernel. We can lock down the mechanisms for upgrading that kernel to require physical access to the phone (e.g., using the docking connector). With this TCB in place, we can introduce a variety of mechanisms to separate potentially hostile apps from one another within the phone, among other benefits. We will discuss this further in Section 4.

3 SME-PED

The Secure Mobile Environment—Portable Electronic Device (SME-PED, pronounced "smeee-pehd") program aims to give U.S. military personnel a device analogous to civilian smartphones, yet also able to make top-secret phone calls and to interact at the secret level with the U.S. military's classified network (SIPRNET) resources. The core design of the current SME-PED smartphones involves four distinct electronics boards: a trusted crypto module, the semi-trusted "black" and "red" compute modules, and the untrusted RF module, which can be physically removed and replaced (e.g., to convert the SME-PED from GSM to CDMA, one would only need to replace the RF module, which clips onto the outside of the phone).

The crypto board does more than just cryptography. It acts in an analogous fashion to a KVM (keyboard, video, mouse) switch, typically used to share a display across multiple separate computers. Here, the crypto module selects whether or not either compute module can see the keys and the microphone, the screen, and so forth. The crypto module has dedicated mode selector buttons, with which the user selects one side of the system or the other. Most importantly, the crypto module has a *trusted path* to tell the user what's going on. On the General Dynamics implementation, this is implemented as a secondary screen, below the keypad. On the L-3 implementation, this is implemented with a segment on the main display that can only be written to by the crypto module.

SME-PED phones have a variety of other features that might be attractive to military personnel, including support for CAC cards (personal identification cards, used throughout the government, which can act as crypto plugin providers for unclassified-but-sensitive email systems). They also meet MIL-SPEC ratings for heat, water, and other environmental factors that would destroy a normal phone.

One curious property of the SME-PED design is that operating system security mechanisms are largely unused. Rather than trusting the operating system, in this case Windows Mobile, to provide separation between "black" and "red" materials, the designers required distinct hardware boards. This certainly simplifies the security analysis, and it also completely contains attacks on the "black" side, such as might originate from a hacked public web server, ensuring that the compromise cannot cross the air gap[1] to the "red" side where it might compromise more important secrets.

To deal with the HVT threat model, SME-PED phones require the user to enter a PIN and will zeroize the internal storage if the PIN entry fails enough times. Furthermore, administrative policies allow for zeroization under various other conditions. All data is encrypted as it's stored, making it more difficult for an attacker to learn valuable information by forcibly extracting the Flash memory chips.

(Remote zeroization and encrypted storage are claimed features of RIM Blackberry products. Comparable features either already exist or are planned for other smartphones.)

[1]There isn't very much actual air in this particular air gap. A broader analysis of the extent to which the military maintains the air gap to its secret and higher materials is beyond the scope of this whitepaper.

Unfortunately, despite all of this engineering, SME-PED phones are fantastically expensive (almost $4000 each). SME-PEDs are heavier and more cumbersome than their civilian counterparts, their battery life isn't very good, and they're largely incompatible with the rapidly growing world of third-party apps, such as have become a major selling point for Apple's iPhone and Google's Android.

Furthermore, even with the SME-PED's sophisticated design, a number of important security concerns remain, at least with regard to the less-trusted "black" side of the phone. If an attacker can compromise the system software, the attacker can then capture audio from the microphone and key presses from the keyboard. When a CAC card is inserted, an attacker could also potentially use it from the black side to generate digital signatures. Of course, switching the phone to the "red" mode would disable all of these features, but a SME-PED may well spend a lot of its time in the "black" mode where such vulnerabilities would be relevant.

An important challenge is to learn lessons from the SME-PED program that can be applied toward civilian phones.

4 Better Smartphones

Unlike the SME-PED, civilian smartphones absolutely must be cheap to manufacture. Given the volumes in which these phones ship, their manufacturers have huge incentives to save money on the margins, so the redundant hardware approach, used in SME-PED systems, would never be taken seriously.

Regardless, civilian smartphones still need appropriate machinery, most likely in software, to separate different apps from interfering with one another. They still need a notion of a trusted path, so users can distinguish whether the app they see is the app they want. And they also need the kinds of remote management and administration that we normally only associate with desktop computers or servers.

4.1 Trusted path

SME-PED handsets achieve trusted path with a dedicated screen (or dedicated area of the screen) that identifies critical security information to the user, such as whether the black or red side of the phone is engaged, and for secure calls, the identity of the remote party. Even if the software on the computer board is compromised, the trusted path display will (hopefully) still say what's going on.

This screen is controlled by the SME-PED's crypto module and cannot be overridden by either compute module. This means that personnel can be trained to pay attention to it. If malicious software were to compromise the black (untrusted) side of the phone and were to pretend that it was showing the user's secure email, the user would hopefully be able to recognize the forgery because the trusted display would clearly indicate which compute module was in charge of the phone.

In addition to the trusted display, the SME-PED also has trusted keys, such as the red/black selector buttons. The wires from these keys (presumably) go directly to the crypto module, eliminating the black CPU's ability to interfere. If the user hits the red button, the user will then be guaranteed to be interacting with the red CPU.

One of our challenges is to get trusted path semantics without having a dedicated screen. The best hook we have is the use of hard-wired buttons, such as the iPhone's central "home" key. On current iPhones, the home key is trapped by the operating system, which will go so far as to kill off an unresponsive app to return the user to the home screen. Microsoft similarly leveraged the Control-Alt-Delete key sequence on PCs as a form of trusted path in its earlier Windows NT versions. To log in, you had to type Control-Alt-Delete first, and no application besides the kernel would ever receive that key. (Microsoft removed this

requirement in Windows XP, since users didn't like it. It also offers less benefit when remote attackers care less about learning the system login password and would much rather learn passwords for online services such as banking.)

4.2 User-facing security policies

For better or for worse, computer users are fantastically unable to respond to security dialogs. To offer an anecdote, a friend was trying to play a DVD on her Windows-based laptop and it wasn't working. I suggested she download "VLC," a popular open-source application that has worked well for me in the past. Her computer's after-market security software then proceeded to generate an impressive storm of messages, including at one point blacking out the screen to present a strident warning. She clicked through all of these without ever bothering to read them. After all, they were getting in the way of what she knew she wanted. A recent study quantified these effects with SSL warnings [11], finding that poorly engineered web browser warnings would be ignored by virtually all users (around 90%) while even the best-designed warnings would only be heeded by half of the users. These studies were conducted with university students, who presumably have more experience with computers than the general population.

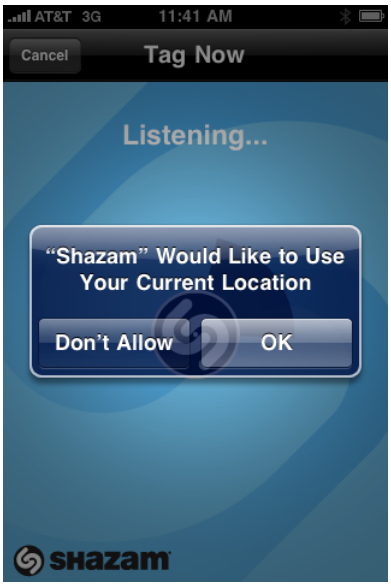

Figure 1: iPhone security dialog for location-based services, shown here with the "Shazam" music recognition service.

The only obvious solution available, unfortunately, is to take these choices out of the hands of users. Every possibly dangerous policy question could ostensibly be answered in advance, in the negative, on behalf of users. This will certainly yield a safer experience, but will also anger users, for whom one of the large attractions of getting a smartphone is to have access to the world of third-party applications. All the benefit of having centrally administered smartphones will be lost if they are sufficiently locked down that users consider them unusable.

Apple's solution is a curious hybrid of enforcing a wide variety of rules in advance, prior to allowing apps into the iTunes Store, along with requiring apps to request permission for more invasive permissions at runtime. Figure 1 shows an interaction with the "Shazam" app, asking for permission to use the GPS location service. Shazam's normal purpose is to listen to ambient music, via the microphone, ship this off to a remote computer for processing, and then tell the user what song it may have recognized. Of course, Shazam will then helpfully offer the ability to buy the music. So why does Shazam need to know where the phone is, physically? Maybe they can offer some social networking feature. Users can tell the world where they were when they heard favorite songs. Alternatively, users can decline to reveal their location to Shazam, as the app works just fine without this information.

Because dialogs such as this are infrequent, the appearance of one will hopefully give some pause to the user, but we should not expect anything close to *all* users to say "no" in circumstances when that would be the preferred answer.

4.3 System-enforced policies

Apple's iPhone security mechanism has two essential flaws. First, it assumes that its app store policing function can prevent dangerous apps from ever reaching users. Second, it assumes that the apps that are present on the phone cannot be exploited. In fact, the latter issue has given rise to the phenomenon of iPhone "jailbreaking," where vulnerabilities are methodically discovered, either in apps or libraries or wherever else, which are then exploited to disable the iPhone's requirement for code to be digitally signed by Apple. At that point, users are free to install apps from anywhere. Jailbreaking methods have changed over time; many now require the iPhone to be tethered to a computer, leveraging the additional power available to the attacker by using the docking connector. (In our LVT vs. HVT taxonomy, this would be an HVT attack.)

An alternate approach is taken by Google's Android. Android phones, under the hood, are running a stripped-down Linux system, much like Apple iPhones run a stripped-down version of the Macintosh OS X system. Android leverages Linux's own protection system, giving each app its own Linux user-id [1], and marking the protection bits in the filesystem such that users can neither see nor edit each others' files. By treating each app as a distinct user, a security compromise in any one app is insufficient to take over the entire Android phone. All of this happens, by design, without requiring any user input. Android apps may still request access to non-default permissions, as with the iPhone, ultimately having similar weaknesses.

For both the iPhone and Android platforms, it's easy to imagine getting the systems administrator in the loop on these permission checks. When a user installs an app, the system could enforce the standard default permissions, yet also send a note to the administrator. The administrator could then make policy exceptions on an app-by-app basis and publish those to every managed phone. This naturally requires having a systems administrator, which will not happen when users are using personal equipment, particularly if they acquired personal smartphones specifically to work around onerous administrative policies elsewhere in their work environment (see the Trends listed in the Introduction).

Consider the issue of what permissions should be allowed by default. Should apps have the privilege to interrogate the address book? One iPhone game developer, Storm8, is being sued over allegedly copying users' address books [3]. Clearly, in an Android-like system, the privilege to access the phone book need not be enabled by default, which would clearly defeat such an exfiltration attack. On the other hand, if too many such privileges were denied, then more exceptions would be necessary to the default security policy in order to enable useful apps which have legitimate reasons to access resources like the address book.

While there will probably never be a clear answer on default, one-size-fits-all security policies, we can imagine building a novel security policy from a rule that Apple requires for the purposes of saving power. Recall that an iPhone is a general-purpose Unix computer. It can certainly allow applications to run in the background, with no visible indication that they are running. Of course, this will slow down the phone and consume more power, so Apple bans this practice as a requirement to get into its online store. Other smartphones, notably Android and HP/Palm phones, allow and encourage background applications.

Apple's power-saving policy could be *enforced* by the operating system kernel, turning it into a *context-sensitive security policy*. When an app is in the foreground, because the user selected it, we can allow the app to access the microphone, speakers, and perhaps also the GPS device. When it's not in the foreground, the OS kernel can forcibly take away these privileges or kill the process altogether. Combined with a trusted path mechanism to select the foreground app (see Section 4.1), this gives users assurance that an app cannot misbehave when it's not visibly running. This, in turn, will give security administrators a greater comfort level in allowing users to use location-aware apps.

4.4 Virtualization and forensics

The concept of virtual machines first appeared on mainframe computers, decades ago, and is now gaining prominence both on desktop machines and in server farms. The idea is that an operating system, such as Windows, does not need to run directly on the hardware. Instead, it can run on a hardware emulator, which then runs on the real hardware. For consumers, this technology, from companies such as VMware and Parallels, makes it possible to run Windows applications seamlessly on a Macintosh computer. For the data center, it allows old software to continue running on new computers, even when the software is incompatible with newer operating systems. Virtualization also creates separation between mutually distrusting entities who share resources. Amazon uses this, to great effect, in its Elastic Compute Cloud[2] service, allowing customers to install any operating system they wish without requiring separate computers for each customer.

Modern smartphones clearly have the necessary resources to support virtualization. How can we leverage this for improved smartphone security? Virtualization allows the smartphone to save and restore its entire state to remote, trusted network services. Think of this as a backup service. If a user loses a phone, the entire contents of the phone, including all of its third-party apps and their data, could potentially be restored in minutes. VMware is already working on such a technology [5]. In addition to backups, we could also consider replicated execution on the phone and on a remote (trusted) server. See, for example, Microsoft's Ripley system [13]. Replicated execution doesn't solve security attacks, but it does bring the execution closer to where it can be monitored and controlled in real time.

At the point where we have off-phone backups, we now have the opportunity for off-phone attack forensics. A forensic analyst, in such cases, has access to the full state of the phone, so even if the phone appeared to be unmodified, stealthy changes could still be discovered. Furthermore, a forensic analyst would have access to large numbers of phone backups, creating interesting opportunities for bulk analysis. If any one phone stands out from the crowd, it may well be under attack and would warrant bringing the phone in for a closer look. Of course, once an attack is detected, a variety of responses can be devised.

With virtualization, we could even imagine implementing software security mechanisms that parallel the hardware security separation techniques used in the SME-PED. This may never be acceptable for secret-level data or voice processing, but it could be used to create a separation between users' "work" uses of the phone and their "home" uses. Making this *usable* would be quite tricky, particularly dealing with thorny issues like the address book. Should it be shared across the two virtual systems? How should we indicate to the user which VM is in use and how can we guarantee that a compromised VM cannot defeat the VM switching mechanism? These issues could possibly be addressed by borrowing some of the trusted path mechanisms in the SME-PED (see Section 4.1) for a regular consumer smartphone, although it may well be simpler and cheaper for users to simply carry two phones.

Virtualization also is one of the key components of secure bootstrapping (see, e.g., Arbaugh et al. [2]), where the hardware verifies integrity of the boot ROM (by comparison to a cryptographic checksum), which then verifies the integrity of the virtualization microkernel, which can then verify the operating system image, and so forth. If at any point in this chain, the verification fails, then access to lower-level resources, such as cryptographic key material, can be denied, thus protecting the confidentiality of data storage, remote authentication, and so forth. These technologies have never really taken off in the commercial world (for example, dealing with the staggering diversity of operating system configurations), but they could play a valuable role in smartphones.

It's important to note that this virtualization approach is actually an excellent solution to Trends 2 and 3 (see the Introduction) to secure the *desktop* environment. If government and military personnel wish to

[2] http://aws.amazon.com/ec2/

access Facebook, Twitter, or the New York Times, these can be separated from internal data by running virtual machines. While a variety of different implementation tactics can be taken, broadly speaking we can create the equivalent of having two separate computers on everybody's desk, one configured for using the internal network, and the other configured for using the external network. Similar approaches are already taken for access to SIPRNET, so the necessary tools and systems administration skills are already available and understood in government and military networks.

4.5 Attack surface analysis

A typical way to analyze the security of a system is to consider the size of its attack surface, which is to say, to consider all the possible ways that an attacker might try to interact with the system. Every software service that accepts network packets is part of the attack surface.

The attack surface of the VM layer is really no different than the security of the regular operating system kernel, assuming the operating system is designed to limit the privileges of regular applications (such as Android but unlike the iPhone). In the HVT threat model, an attacker with physical access to a phone will ultimately be able to replace the code running on it. In the LVT threat model, however, either a VM layer or a traditional operating system kernel running above it, need only be designed to resist external attacks, via hostile data transmitted over the network.

Ultimately, an attacker may try to attack an existing, trusted app (whether built-in or third-party), an attacker may manage to get a malicious app installed, or an attacker may choose to go after the kernel or VM (if present). The separation that comes from Android's model (see Section 4.3) would seem like a good way to contain attacks against apps. Defeating malicious apps, particularly if they appear to be legitimate ones, is tricky. This requires limiting the ability of illegitimate apps to pretend to be legitimate ones. Apple solves this by requiring all apps to be digitally signed by its online store and by enforcing that it doesn't endorse apps that appear to be the same as others. Absent centralized control, such as where Android allows apps to come from anywhere, it becomes a UI problem to ensure that a malicious app cannot be confused for something else combined with the risk of social engineering attacks (e.g., "spear phishing") which try to convince users to download and install things which they shouldn't.

The attack surface also includes whatever mechanism is in place to do software upgrades to the phone operating system. This includes "official" upgrades, which may be pushed by phone carriers over the air. But should phone carriers be trusted? In July 2009, Etisalat, a phone provider in the United Arab Emirates, sent out a notice to Blackberry customers inviting them to download an update which turned out to be spyware [7, 9]. If the phone carrier cannot be trusted, then who can? This particular update was really just a regular app that wired itself deeply into the Blackberry system, to spy on the user, and send that data back to a central server [4], so it's really more in the above category of a malicious app than in the category of an attack on the operating system itself. Nonetheless, the concern remains. Should we insist that OS upgrades be digitally signed by the original vendor? That would limit the ability of a malicious phone carrier to monkey with a phone, but the crypto would need to be reasonably strong, as it would be an attractive target for concerted nation-state cryptanalytic attacks. For contrast, the digital signature scheme used by Texas Instruments for its calculators was broken [14], enabled by virtue of the 512-bit signature scheme used being far too weak [10].

If we want to protect LVT phone users against attack, then we can consider security policies that require all installations, whether to replace the operating system or its apps, to go through the docking connector. On the one hand, this eliminates the Etisalat attack and others like it. On the other hand, the modern personal computer isn't exactly a trustworthy place, either, and PC-based malware could well be engineered to attack the smartphone when it's plugged in. This suggests that a combination of strong digital signatures plus the

use of the physical docking connector may be required in combination.

5 Recommendations

This think piece opened by considering several trends in government, military, and corporate computer security policies which are pushing users away from doing personal activities on their work equipment. Of course, this just pushes such activities to personal computers and smartphones without the scrutiny and administrative controls available on professionally managed computers.

A better solution is to harden the users' equipment, whether using virtualization technologies to separate users' personal and work computing spaces, while still running in a single computer. Furthermore, replacing old and insecure web browsers such as Internet Explorer 6 with newer, more robust web browsers, such as Google's Chrome [8], would significantly improve the security posture for web surfing. With such mitigations in place, it becomes safer and preferable to have users visit potentially hostile web sites from the inside, rather than using personal equipment to visit such sites from home.

Smartphones, meanwhile, have all the computational power of full-blown desktop computers, yet they fit in our pockets and go with us everywhere, opening up a variety of new security concerns. If a phone is compromised, it becomes a powerful electronic eavesdropping device. While banning such phones would appear to be the answer, and many government and military sites forbid any phones from being brought inside, they nonetheless serve as attractive targets for remote exploitation. With improved software architectures on these phones, they could be centrally managed, allowing professional systems administrators to do forensic analyses, detect attacks when they happen, and respond in a timely fashion. None of that is possible when users are using personal equipment.

Many of the technologies discussed here are fairly close to technologies that are already shipping in commercial products, but there a world of difference between "close" and "secure." By virtue being such a large purchaser, the government can have a significant influence on the design and engineering of smartphones. If commercial smartphone engineers were brought in and briefed on how close they are and how they could do better for all of their users without costly hardware modifications, this could have a real impact.

While much of this think piece has focused on fairly practical and short-term issues, some long-term research challenges are also worth considering. Imagine a database with millions of smartphone backups, all in one place. How should this be analyzed? How, exactly, does this enable the administrator to detect attacks? Tools need to be developed that can distinguish between normal variances from one phone to the next and identify attacks without generating too many false alarms.

Furthermore, if we require that apps be digitally signed before they are installed, then the signing authority can apply significant static analysis (or binary rewriting for dynamic analyses) to the apps. Can we prove that apps are well behaved? Can we even formalize what "good behavior" should be? The IARPA "STONESOUP" research program[3] is pursuing exactly this problem set.

Lastly, when humans are being required to take actions with security ramifications, the next immediate question should be to measure how often they can succeed. Human subject experiments can measure this, can determine what sort of training can have the best bang for the buck, and can consider design variations to determine how best to structure the interface. Trusted path mechanisms are essential to the security of smartphones. Identifying whether users respond properly when the trusted path indicators show there's a problem is essential to understanding whether the system, at the end of the day, is actually going to be secure.

[3] http://www.iarpa.gov/solicitations_stonesoup.html

References

[1] Android Developers. *Security and Permissions*, Oct. 2009. http://developer.android.com/guide/topics/security/security.html, downloaded November 2, 2009.

[2] W. A. Arbaugh, D. J. Farber, and J. M. Smith. A secure and reliable bootstrap architecture. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 65–71, Oakland, CA, May 1997.

[3] R. Beschizza. iPhone game dev accused of stealing players' phone numbers. *Boing Boing*, Nov. 2009. http://www.boingboing.net/2009/11/05/iphone-game-dev-accu.html.

[4] C. Eng. *BlackBerry Spyware Dissected*. Veracode, July 2009. http://www.veracode.com/blog/2009/07/blackberry-spyware-dissected/.

[5] A. R. Hickey. Interop: VMware to bring virtualization to smartphones. *ChannelWeb*, May 2009. http://www.crn.com/storage/217600096.

[6] Marine Corps. *Immediate ban of Internet social networking sites (SNS) on Marine Corps enterprise network (MCEN) NIPRNET*, Aug. 2009. MARADMIN Active Number: 0458/09, http://www.marines.mil/news/messages/Pages/MARADMIN0458-09.aspx.

[7] B. Ray. BlackBerry update bursting with spyware. *The Register*, July 2009. http://www.theregister.co.uk/2009/07/14/blackberry_snooping/.

[8] C. Reis, A. Barth, and C. Pizano. Browser security: Lessons from Google Chrome. *ACM Queue*, June 2009.

[9] Research in Motion. *App Remover for removing Etisalat's "Registration" application on BlackBerry smartphones*, July 2009. http://na.blackberry.com/eng/ataglance/security/regappremover.jsp.

[10] B. Schneier. Texas Instruments signing keys broken. *Schneier on Security*, Sept. 2009. http://www.schneier.com/blog/archives/2009/09/texas_instrumen.html.

[11] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying wolf: An empirical study of SSL warning effectiveness. In *18th USENIX Security Symposium*, Montreal, Canada, Aug. 2009.

[12] A. Vance. Times web ads show security breach. *New York Times*, Sept. 2009. http://www.nytimes.com/2009/09/15/technology/internet/15adco.html.

[13] K. Vikram, A. Prateek, and B. Livshits. Ripley: Automatically securing Web 2.0 applications through replicated execution. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*, Chicago, IL, Nov. 2009.

[14] M. Vincent. TI-83 Plus OS signing key cracked. *ticalc.org*, July 2009. http://www.ticalc.org/archives/news/articles/14/145/145154.html.