Defending Against Sensor-Sniffing Attacks on Mobile Phones

Liang Cai
University of California
Davis, California
lngcai@ucdavis.edu

Sridhar Machiraju
Sprint Applied Research
Burlingame, California
machiraju@acm.org

Hao Chen
University of California
Davis, California
hchen@cs.ucdavis.edu

ABSTRACT

Modern mobile phones possess three types of capabilities: computing, communication, and sensing. While these capabilities enable a variety of novel applications, they also raise serious privacy concerns. We explore the vulnerability where attackers snoop on users by sniffing on their mobile phone sensors, such as the microphone, camera, and GPS receiver. We show that current mobile phone platforms inadequately protect their users from this threat. To provide better privacy for mobile phone users, we analyze desirable uses of these sensors and discuss the properties of good privacy protection solutions. Then, we propose a general framework for such solutions and discuss various possible approaches to implement the framework's components.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Information flow controls, Invasive software*

General Terms

Design, Security, Human Factors

Keywords

privacy, mobile, sensor, sniffing, microphone

1. INTRODUCTION

Unlike mobile devices in the past, which were designed for the sole purpose of voice-based communication, today's phones are powerful devices that can communicate, compute, and sense. The sensing capabilities of mobile phones come from the audio, video, and location sensors in the form of microphones, cameras, and GPS receivers. While these sensors enable a variety of new applications, they can also seriously jeopardize user privacy. In particular, if a mobile device is compromised, an adversary can not only access the data stored in the device but also record all of the user's actions by stealthily sniffing on the sensors.

MobiHeld'09, August 17, 2009, Barcelona, Spain.

We focus on threats to users' privacy due to unauthorized sniffing on mobile phone sensors. These threats are different from the more traditional attacks on user privacy on PCs. Those attacks aim at (1) accessing private data or (2) eavesdropping on users' operations (e.g., key loggers). The first type of attacks can be defeated by proper file access control policies or encryption. The second type of attacks is effective only when the user is interacting with his PC. In contrast, appropriate access control on sensors often depends on the context, so a static access control policy with no regard to the context, which is typical on file systems, is inadequate. Also, these attacks work even when the user is not interacting with the mobile phone. As long as the phone is within the proximity of the user, which is often the case, the attacker can continuously snoop on the user's activities. Such snooping can also help attackers compromise other computing devices. For example, the attacker can listen to the acoustics of the keyboard to infer typed passwords [1].

Although PCs can also be equipped with sensors, the sensor-sniffing problem is much more serious on mobile devices. On PCs, most sensors are optional, *add-on* peripherals. They are not universally available, are not used by most applications, and can be turned off without affecting most tasks. In contrast, microphones are universally available and indispensable on all mobile phones; cameras and GPS receivers are moving in this direction as well. Mobile applications use and depend on sensors more extensively. Moreover, users tend to carry mobile devices wherever they go. Hence, sensor-sniffing attackers have many more opportunities to compromise the privacy of mobile users than PC users.

Prior work has raises privacy concerns about mobile phone sensors, primarily in the context of location sensing where location is either read from GPS sensors in mobile phones or inferred from other sources such as nearby cells [2, 3]. Solutions to mitigate this problem have mainly focused on defining security policies [4, 5] or privacy rules [6]. Some commercial software allows tracking employees and children [7, 8]. To our knowledge, there has been little focus on the other sensors (particularly microphones and cameras, which are arguably the most important sensors) on mobile phones except for scattered reports in non-scientific literature [9].

We start our exploration of sensor-sniffing attacks by developing an appropriate threat model. Under this threat model, we show that current mobile phone platforms inadequately protect users. We use existing applications to understand and classify legitimate uses of sensors. The complexity

and diversity of such uses pose significant challenges in developing a single mechanism to detect unauthorized sensor sniffers. Instead, we develop a general solution framework and describe how we could implement its various components. We take advantage of various unique properties of the mobile platform to design novel mechanisms for these components. For instance, we could use the context inferred from sensors to enforce context-aware access control policies.

Our contributions are threefold. First, we examine the privacy implications of powerful sensing capabilities in mobile phones. Second, we demonstrate the significant challenges in alleviating the sensor-sniffing problem. Third, we develop a general framework for preserving privacy and identify a few promising first-cut approaches for its components. Even though we are far from achieving a complete privacy-preserving solution for mobile phones, our work aims at spurring further research into this important area. The continued popularity of mobile phones will be seriously jeopardized if users start to view them as untrustworthy.

2. PROBLEM SCOPE

2.1 Problem Definition

We consider attacks that violate user privacy by sniffing on the sensors on mobile phones. Typical sensors include the microphone, camera, and GPS receiver.[1] Focusing on these sensors, particularly microphone and camera, we illustrate that sensor-sniffing attacks pose challenges that are quite different from those posed by general malware. We also do not consider attacks that sniff on traditional input devices, such as the keyboard and mouse. The reason is that these attacks have been investigated extensively on desktop computers, and we expect the defenses developed apply to mobile phones as well. We also do not consider attacks that steal confidential files for the same reason as above.

2.2 Threat Model

We define the threat model of sensor-sniffing attacks as follows. The threat model defines the capabilities of attackers, which are necessary for evaluating our proposed solutions.

- We assume that attackers are able to install malicious software on mobile devices. The attackers can achieve this by exploiting software vulnerabilities (e.g., drive-by download) or tricking users into installing untrusted code.
- We assume that attackers have no physical access to the compromised mobile devices and can receive the captured sensor data only via voice or data channels, such as outgoing phone calls, SMS, MMS, and TCP connections.
- When we investigate defense approaches, we will discuss how to implement the defense mechanisms in the operating system, on the assumption that attackers cannot compromise the operating system. However, this assumption is not a fundamental requirement of our approach. If the operating system is vulnerable, we could move the defense mechanisms into the virtual machine monitor where available or the firmware.

[1]Other possible sensors on mobile phones include accelerometers, which are increasingly popular, and other more esoteric sensors, such as thermometers and barometers. Attackers can sniff on these sensors to violate users' privacy as well. However, for a focused discussion, we target the most prevalent sensors today — microphone, camera, and GPS.

3. CURRENT USES AND PROTECTION OF SENSORS

In this section, we describe how legitimate applications use mobile phone sensors in different ways and why they can complicate protection mechanisms. Then, we show that popular smart phones protect their sensors inadequately.

3.1 How Applications Use Sensors

To prevent malicious use of sensors, we need to understand how legitimate applications may use sensors. We classify legitimate uses of sensors into three categories, based on how prominent a role sensors play in the applications.

3.1.1 Dominated by Sensors

In this category, the main function of the application is to capture the input of the sensor. For example, the microphone provides input to the telephony, VoIP, and voice recorder applications; the camera provides input to photo and video capture applications. Such an application turns on a sensor at start up and turns it off at completion. The user is aware that the application is using the sensor continuously.

3.1.2 Supported by Sensors

In this category, sensors provide auxiliary input to applications, but the main function of the application is not to capture sensor input. For instance, a voice-dialing application reads the user's voice from the microphone, recognizes the phone number, and then dials the number. [10] uses camera-equipped mobile phones to interact with real-world objects. Some applications may also send the captured data back to a remote server. For example, the Android application CompareEverywhere[11] can capture the barcode image of a product and compare its price with those in nearby stores using an online database. In these cases, the application need not turn on the sensor throughout its lifetime, but the user knows when the sensing starts and ends.

3.1.3 Using Context Provided by Sensors

In the previous two categories, the user initiates the sensing by the applications. By contrast, context-aware computing [12, 13] automatically detects the user's context by sensing continuously. For example, [14] proposes to use the camera as a light sensor. [15] describes how to use the microphone to detect the ambient noise level to adjust the ringer volume accordingly. Recently, [16] demonstrated how to infer the distance of two smart phones using their speakers and microphones. In all these applications, sensors provide contextual information specific to the environment. As such, users may not be aware that the sensor is recording continuously.

3.2 Why a Hardware Switch Won't Work

One might propose a simple hardware switch to turn on and off sensors. It might work well with the first category of applications. Since they turn on the sensors throughout their lifetime, we could combine the hardware switch with

the buttons that start the applications. The advantage of this solution is that no extra work is required from users. However, this approach does not work well with the second category of applications. Since the sensor need not be turned on throughout the application's lifetime, the hardware switch cannot be combined with the application-start buttons. Therefore, the phone would need extra buttons. Moreover, it requires extra work from the user – e.g., switching on the microphone to start voice-dialing and switching it off when he is finished – which can be annoying. Finally, it would be infeasible to design a hardware sensor switch for the third category of applications, since they require the sensors to remain on at all times.

3.3 Limitations of Current Systems

Current mobile operating systems, such as Windows Mobile[17], Symbian[18], BlackBerry[19], and Google Android[20], provide certain mechanisms for protecting sensors, but these mechanisms are inadequate.

Certification: Among today's systems, the most widely-used security solutions are based on certification. These solutions encourage users to install and use applications only if they have been certified by a trusted source. Although widely adopted by mainstream mobile platforms, the limitations of certification are obvious: (1) It merely raises the bar for malware developers without providing real security assurance. Moveover, even if the certification authority can verify that an application satisfies its privacy policy, its privacy policy may differ significantly from the user's desirable policy. (2) Certification can be circumvented when users are tricked into installing malware bundled with an otherwise compelling application such as a game. (3) Applications are often certified based on organizational trust relationship rather than technical verification.

Reference Architecture: Reference architectures[21, 22] apply traditional OS security mechanisms — such as sandboxing, run-time monitoring, and integrity verification — to the mobile platform. For example, Google Android requires each application to list all the privileges that it needs (including accessing hardware and network connection) in a manifest file and detects any violations in runtime. However, this does not solve the sensor-sniffing problem because it continues to rely on user knowledge and diligence to grant/deny access.

To help us understand the ease of writing sensor-sniffers, we experimented with one of the above mobile platforms. We used a mobile smartphone with built-in Assisted GPS module and a 2.0 megapixel camera. We easily developed a program that periodically records 30 seconds of sound, takes a picture, and reads location information from the GPS. The program stores the recorded data in a file and later uploads it to an FTP server. The phone notifies the user of this program's activities only when the program dials up to establish a network connection. However, the program could avoid suspicion by waiting for another program to establish a network connection and then using that connection to send out the recorded data.

Recent work [23] has also shown the feasibility of sniffing the video sensor by building a video capture malware with specific trigger algorithm and infection methods. It shows that such malware can be implemented with limited use of power, CPU, and memory, thereby making detection hard as well.

4. DESIGN OF A DEFENSE SYSTEM

Given the variety of ways in which legitimate applications can use sensors, we believe that a single solution cannot be a complete defense. Instead, we believe that a general framework that can accommodate several design choices is more appropriate. Before we present the framework and its key components, we first discuss the properties desired for an ideal solution.

4.1 Desirable Properties

An ideal solution must reliably prevent malicious programs from sniffing phone sensors without imposing unacceptable burdens on users. Specifically, it must possess the following properties:

- **Security:** The solution must be able to prevent malicious programs from reliably accessing protected sensors.
- **Usability:** Ideally, the solution should require no user intervention. This way, the security solution incurs no usability cost to the user. If user intervention is unavoidable, the user should be able to make informed security decisions and should not have to make such decisions too often. The decisions should not disrupt the user's work flow.
- **Backward and Forward Compatibility:** The solution should require no or minimal modification to existing applications. Existing applications that access sensors should continue functioning. Since we cannot predict future applications, we should also avoid restricting how future applications may use the sensors. Therefore, the solution should allow diverse ways of using the sensors.
- **Performance:** The solution should have small overhead and should not considerably degrade the performance of the OS or applications.
- **Versatility:** Mobile devices have a plethora of hardware, software, and user interfaces. The solution should apply to these various devices.

4.2 Framework of the defense system

Our proposed framework consists of three modules: *policy engine*, *interceptor*, and *user interaction*. (Figure 1). Since our threat model excludes the case where attackers can physically access the mobile device, the framework either protects sensors from unauthorized access or prevents sensory data from leaving the device via the network. The policy engine determines whether to allow each access, based on the input from user interaction and application monitoring/profiling. The interceptor enforces the decision by the policy engine. Next, we discuss how we can implement these modules using known mechanisms.

4.2.1 Policy Engine and Application monitoring

The framework will have good usability if its policy engine makes its decisions based mainly on application monitoring and profiling without requiring much user intervention. We explore several such policy options.

Whitelisting and blacklisting.

We could whitelist all the applications that legitimately require access to the sensors. For example, the telephony application and voice dialing application require access to the

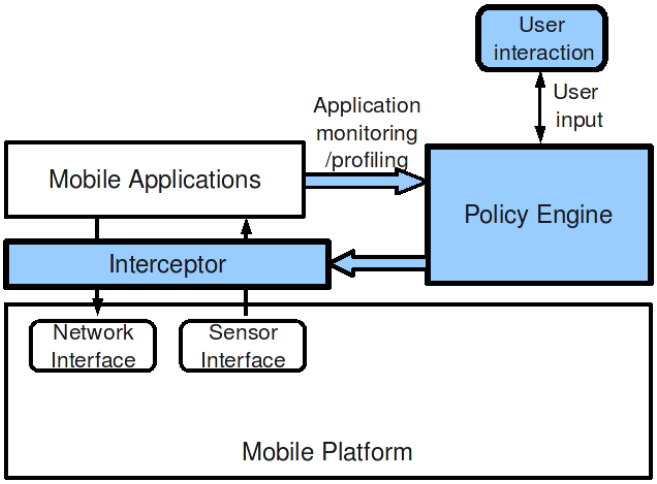

Figure 1: The framework for defending against sensor-sniffing attacks consists of three modules: policy engine, interceptor, and user interaction.

microphone and the navigator application requires access to the GPS sensor. Alternatively, we could blacklist all the known malicious applications. The lists may be created and maintained by the device manufacturer, OS provider, network operator, and user. Both the whitelisting and blacklisting approaches run into problems when the user installs new applications because they require the user to decide whether to trust the new applications, which is a daunting task. We may safely whitelist applications in Section 3.1.1 and 3.1.2. For the applications described in Section 3.1.3, we may have to use the more sophisticated approach below.

Information flow tracking.

The threat model in Section 2.2 assumes that the attacker cannot access the mobile device physically. Therefore, the attacker must retrieve sniffed sensor data via the network. This implies that we can allow all the applications that do not access the network to access the sensors. For instance, an application that adjusts the ring volume according to the background noise level does not access the network, so we can allow it to access the microphone.

The above approach, however, cannot prevent two colluding applications from sending out sensory data. For example, one application reads from the sensor and writes the data into a file, and then the other application reads the file and sends the data to the network. To prevent such an attack, we would need information flow tracking. We can treat all the data read from the sensors as tainted, track the flow of tainted data, and prevent tainted data from leaving for the network.

However, taint-based information flow tracking has its own shortcomings. The biggest one is the performance degradation. Also, since dynamic analysis cannot track implicit information flow accurately [24], a malicious program could defeat tainting tracking. Finally, some applications may need to access both the sensor and the network legitimately. For example, an application may take pictures and upload them to a server.

4.2.2 *User interaction*

Policy decision without user intervention is desirable but sometimes infeasible. In this case, the system needs to notify the user and ask for the user's decision. Note that the design of user interaction on the mobile phone is different from other systems.

User authorization.

The system can ask the user if he authorizes an access to a sensor. Traditionally, researchers do not expect ordinary users to make correct access control decisions, because these users do not understand the system enough. In contrast, we argue that we can rely on ordinary users to make informed decisions on access control to certain sensors. This is because although ordinary users may not understand operating system resources, such as the file system and sockets, they do understand the purposes of the microphone, camera, and GPS device. Therefore, in many cases, an ordinary user can decide whether an application requires legitimate access to a sensor (In contrast, ordinary users may not be able to answer whether application X needs to access file Y.) Particularly, it should appear obvious to users that applications in Section 3.1.1 and 3.1.2 need legitimate access to sensors. The system can ask the user for access control decisions via common approaches such as dialog boxes. The system should take care to prevent malicious applications from spoofing such dialog boxes.

Sensor-In-Use notification.

Traditionally, when a system allows a malicious application to access sensitive data, the confidentiality of the data may be violated immediately. However, in the case of sensor-sniffing attacks, the consequence of allowing malicious applications to access sensors is not as severe. This is because when the malicious application acquires permission to access a sensor, there is no confidential data at the sensor yet (we assume that the operating system does not cache sensory data). Therefore, when a user cannot make an informed access control decision, the system can allow the access but notify the user continuously, e.g., via a flashing icon or message box on the screen or intermittent beeps. This notification keeps the user aware that an application is reading from a sensor so that the user might decide to avoid sensitive speech, view, or locations. Some PC operating systems notify users of sensor activities, such as WebCam. These notifications are typically unobtrusive and are therefore easily overlooked by users. As such, this solution is inadequate for mobile phone sensors, where the privacy violation caused by sensor sniffing is much more severe. We need further research on human computer interaction to find a proper tradeoff between unobtrusiveness and privacy protection.

4.2.3 *Interceptor*

The interceptor interposes between the application and the sensors, and/or between the application and the network. It enforces the policy decision on accessing sensors and the network. When the policy prohibits an application from accessing the network, the interceptor simply denies such access. In contrast, when the policy prohibits an application from accessing the sensor, there are two simple options for enforcement.

The first option is **locking**. On OSes where the sensor interface provides exclusive access, such as Windows Mobile where both the microphone and camera can be accessed exclusively by an application, we can lock the sensors. For example, the interceptor can be a daemon program that opens the protected sensor. When a legitimate program requires access to the sensor, the daemon closes the sensor to relinquish its lock. The advantage of this approach is that it requires no modification to the OS. The disadvantage is that it may be susceptible to a race condition, where both a legitimate and malicious application compete for the released sensor.

To avoid the race condition above, we can use the second option of **blocking**. We modify the access control to the sensors so that the OS may block access to protected sensors until the user authorizes the access. The disadvantage is that if the user mistakenly denies access or is unavailable for making a decision, the sensory data is lost. Such false positives may be unacceptable where the sensory data is critical.

The above discussion applies traditional lock-and-release semantics to sensor access control. However, it may not be the only or the optimal solution. In the next section, for example, we will show that we could control access based on user context, such as the location. We will also discuss a semantics based on fail-safe encryption of sensory data.

5. NOVEL SOLUTIONS BY LEVERAGING MOBILE PLATFORM

In Section 4.2, we discussed how we can use known approaches to defend against sensor-sniffing attacks. Each approach satisfies only some of the desirable properties in Section 4.1. This is unsurprising, since we face similar challenges as the general problem of malware detection.

However, there are important distinctions between sensor-sniffing attacks and general malware attacks. With sensors on mobile platforms, for example, when the user cannot decide whether a program is trusted, he can let the system notify him if the program accesses the microphone (so that he will not talk about sensitive information when the notification is on). By contrast, this *allow but notify* approach is inappropriate for defending against general malware attacks because once the system allows the malware to read a file, the confidentiality of the file may be violated immediately. The unique aspects of mobile platforms also provide us with opportunities to investigate novel defense mechanisms. We provide three such examples.

Context-aware application profiling.

A unique aspect of the mobile platform is that it can look at the user's context (environment or intention) to determine whether to allow an untrusted program to access sensors. By contrast, such context is largely irrelevant to the general malware defense problem, e.g.:

- **Location tagging:** If the mobile device is equipped with GPS, the user can tag certain locations with specific policies. For example, the user may tag a conference room with the policy that no application is allowed to use the microphone or camera.
- **Activity inference:** One useful context information in detecting sensor-sniffing is user activity. For instance, after a legitimate voice application opens the microphone device, we expect the user to speak soon. By contrast, when a malicious application opens the microphone device surreptitiously, there may not be voice activity. Therefore, the presence of voice soon after the opening of the microphone likely indicates that the user is aware of the use of the microphone. Using voice activity detection (VAD) technology[25], we need not notify users of such legitimate use of the microphone.

The advantage of the context-aware approach is that it requires no user interaction. However, this approach, especially activity inference, is often imprecise. Moreover, context-based access control may apply only to certain sensors such as the microphone. For example, it would be difficult to determine in what context to allow a navigator application to access the GPS sensor.

Leveraging existing user interaction.

A well-understood user interaction mechanism in phones is that of the voice calling application. The user presses the *talk* button to initiate a phone call and the *hangup* button to terminate the call. We could use these two buttons as the user's authorization and de-authorization to access the microphone. In this case, the system requires no extra work from the user and has excellent backward compatibility.

Ensuring both security and reliable sensory data capture through encryption.

From the security point of view, when the user does not know whether an application can access a sensor, he should deny such access. However, when this decision is wrong, the sensory data is lost forever. This dilemma might encourage users to always authorize access, which is a security risk. To ensure both security and reliable capture of sensory data, we could use encryption. In this approach, all applications can access the sensors; however, the sensory data is encrypted unless the OS determines that the application is benign. If a legitimate application does not receive authorization at the time when it reads the sensory data, it may save the encrypted sensory data so that the user may decrypt the data and re-feed it to the application later. The disadvantage of this approach is that the application may need to be rewritten if it desires to save encrypted sensory data. Note that if the application receives proper authorization when it accesses the sensor, no modification to the application is necessary.

6. CONCLUSION

With the rapid proliferation of mobile devices and applications, the problem of privacy violations based on sensors of mobile devices is likely to become serious. We used existing and emerging applications to highlight the challenges facing the detection and elimination of such sensor-sniffing malware. We found that existing mobile operating systems provide little protection against such attacks. We enumerate the requirements of an ideal privacy-preserving solution and proposed a framework that consists of three key modules: user interaction, policy engine, and interceptor. For each module, we explored different mechanisms and discussed their advantages and limitations. In particular, we discussed several novel mechanisms, such as context awareness, that the mobile platform enables.

Though we proposed several first-cut solutions, the problem of designing practical and effective countermeasures is far from being solved. All the approaches we discussed in the paper have their own drawbacks. Approaches based on user authorization are based on the assumption that the use of the sensors should be perceptible by users. However, new sensors and applications might violate this assumption. Approaches based on exploiting existing user interactions (e.g., phone ringing) may not work with applications that are always on.

Based on our previous discussion, we believe that it is very challenging, if not infeasible, to eliminate the sensor-sniffing threat completely. However, we are cautiously optimistic that a user-friendly solution, which works effectively in most circumstances, can be devised. Designing such a solution requires research into mobile user behavior, algorithms for automatic context inference, and operating system primitives such as information flow.

7. REFERENCES

[1] Li Zhuang, Feng Zhou, and Doug Tygar. Keyboard Acoustic Emanations Revisited. In *Proc. of ACM CCS*, November 2005.

[2] Maya Gadzheva. Privacy concerns pertaining to location-based services. *Int. J. Intercultural Information Management*, 1(1), 2007.

[3] Calvert L. Bowen and Thomas L. Martin. A survey of location privacy and an approach for solitary users. *Hawaii Intl. Conf. on System Sciences*, 0:163c, 2007.

[4] Norman Sadeh, Jason Hong, Lorrie Cranor, Ian Fette, Patrick Kelley, Madhu Prabaker, and Jinghai Rao. Understanding and capturing peopleÂŠs privacy policies in a people finder application. *Proc. of the Ubicomp Workshop on Privacy*, 2007.

[5] Jason Cornwell, Ian Fette, Gary Hsieh, Madhu Prabaker, Jinghai Rao, Karen Tang, Kami Vaniea, Lujo Bauer, Lorrie Cranor, Jason Hong, Bruce McLaren, Mike Reiter, and Norman Sadeh. User-controllable security and privacy for pervasive computing. In *Proc. of HotMobile*, 2007.

[6] John Morris and Jon Peterson. Who's watching you now? *IEEE Security and Privacy*, 5(1):76–79, 2007.

[7] Trackem. `http://www.solutionsintomotion.com/`.

[8] Wherify Wireless. `http://www.wherifywireless.com`.

[9] Eric Bangeman. FBI using cell phone microphones to eavesdrop. December 2006. `http://arstechnica.com/news.ars/post/20061203-8343.html`.

[10] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In *Advances in Pervasive Computing*, pages 265–271, 2004.

[11] Jeffery Sharkey. CompareEverywhere. `http://compare-everywhere.com/`.

[12] Hans w. Gellersen, Albrecht Schmidt, and Michael Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. volume 7, pages 341–351, 2002.

[13] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[14] Alex Olwal. Lightsense: enabling spatially aware handheld interaction devices. volume 0, pages 119–122, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[15] Chris Mitchell. Adjust Your Ring Volume For Ambient Noise. In *MSDN Magazine*, 2007. `http://msdn.microsoft.com/en-us/magazine/cc163341.aspx`.

[16] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *SenSys*, 2007.

[17] Security Model for Windows Mobile 5.0 and Windows Mobile 6. February 2008. `http://go.microsoft.com/fwlink/?LinkId=118667`.

[18] Symbian Platform Security Model. August 2008. `http://wiki.forum.nokia.com/index.php/Symbian_Platform_Security_Model`.

[19] BlackBerry Enterprise Solution: Security Technical Overview. 2008. `http://www.blackberry.com/knowledgecenterpublic/livelink.exe/BlackBerry_Enterprise_Solution_Security_Technical_Overview.pdf`.

[20] Security and permissions in android. 2008. `http://code.google.com/android/devel/security.html`.

[21] Lieven Desmet, Wouter Joosen, Fabio Massacci, Katsiaryna Naliuka, Pieter Philippaerts, Frank Piessens, and Dries Vanoverberghe. A flexible security architecture to support third-party applications on mobile devices. In *Proc. of ACM workshop on Computer security architecture*, 2007.

[22] Xinwen Zhang, Onur Aciicmez, and Jean-Pierre Seifert. A trusted mobile phone reference architecturevia secure kernel. In *Proc. of STC*, 2007.

[23] Nan Xu, Fan Zhang, Yisha Luo, Weijia Jia, Dong Xuan, and Jin Teng. Stealthy video capturer: a new video-based spyware in 3g smartphones. In *Proc. of WiSec*, 2009.

[24] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE JSAC*, 21(1):5–19, 2003.

[25] J. Ramírez, J. M. Górriz, and J. C. Segura. *Voice Activity Detection: Fundamentals and Speech Recognition System Robustness*, pages 1–22. I-Tech Education and Publishing, Vienna, Austria, 2007.