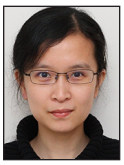
Privacy-Preserving Experimentation with Sensibility Testbed

YANYAN ZHUANG, ALBERT RAFETSEDER, AND JUSTIN CAPPOS



Yanyan Zhuang is a Research Professor at the Computer Science and Engineering Department at New York University. Her current research interests include fault diagnosis of distributed systems and the design and implementation of network testbeds for mobile devices. yyzh@nyu.edu

Albert Rafetseder is a Research Professor at the Computer Science and Engineering Department at New York University. He is the current technical lead for the Seattle Testbed project and takes an interest in large-scale application-layer network measurements. albert.rafetseder@univie.ac.at

Justin Cappos is an Assistant Professor in the Computer Science and Engineering Department at New York University. Justin's research philosophy focuses on improving real-world systems, often by addressing issues that arise in practical deployments. jcappos@nyu.edu

Recent privacy breaches and security break-ins of mobile systems have raised concerns about using mobile devices like smartphones and tablets [1]. As a result, many users are aware that running apps on their smartphones can increase privacy risks. On the other hand, the data from the enormous number of smartphones, if used properly, can be of tremendous value to the research community. Is there a way to safely do research on these devices without rendering them vulnerable? We explain about how our project may help both researchers and volunteers.

Ever wondered what science could achieve if any researcher can get data from other people's smartphones? Imagine that we would simply write a few lines of code, fire it up on a number of strangers' phones, and within minutes we would know where the dead spots of our mobile data plans are. We could also have a zero-cost navigation system when no GPS or any other location services are available; we could achieve this by establishing a Bluetooth connection with a neighboring device and get the location data from it. If we constantly monitor accelerometer data on mobile devices, we can detect vibrations within the frequency and intensity range of seismic waves, and assist distributed earthquake detection. These all sound fantastic, except that who would let us get data off their devices? Our friends and family would probably trust us. Other people? Not so much.

The privacy and security challenges on mobile devices have increased dramatically over the years. Although having apps post tweets to a user's Twitter account without asking for permission is seriously off-putting [3], hacked apps that let criminals break into an individual's bank account are clearly detrimental [4]. Running code to collect data from smartphones is much more complex than it sounds. We not only need to ensure the security of a device so that the code that does the data collection cannot damage or hack into the device, but we also must protect the privacy of a device owner so that the code cannot eavesdrop on phone conversations, steal passwords, etc.

We introduce Sensibility Testbed [5], a smartphone testbed that allows researchers to run code and perform measurements on others' smartphones for research purposes. It ensures the security of user-owned devices and the privacy of user-generated data. The usage model of Sensibility Testbed is unique in that it manages how device owners make their devices accessible to different research communities without putting their devices at risk. Meanwhile, it offers technical resources that allow researchers to collect data from remote mobile devices without impairing the device owner's privacy. As an added bonus, different research groups can pick, choose, share, and reuse each others' user base.

Yet Another Testbed?

While the rich set of sensors on mobile devices can provide useful data sets for research, today's security and privacy issues have created many obstacles to collecting data and sharing them among mobile devices. Note that in this work, sensors are broadly defined as the hardware components that can record phenomena about the physical world, such as WiFi/

cellular network, GPS location, movement acceleration, etc. The challenges of collecting sensor data are twofold: first, sensor data from mobile devices can reveal device owners' personal information and result in privacy breaches; second, potential bugs, sometimes inadvertent ones, in a research experiment can damage end users' personal devices and cause security issues. Collecting data using untrusted programs poses significant challenges for both device owners and bystanders. To run code safely on a stranger's smartphone without revealing this stranger's privacy sounds like a fantasy, or at least, mission impossible.

You are probably wondering, aren't there plenty of network testbeds out there, and don't they already resolve these issues? The problem with existing testbeds is that they do not yet have a systematic way to protect device owners' security and privacy. To lower potential risks, many smartphone-related testbeds choose to recruit participants from a trusted group, such as students, colleagues, and friends. For example, PhoneLab (https://www.phone-lab.org/) provides a platform for people to run Android apps on their participants' smartphones and log data. PhoneLab recruits participants by giving them an Android device and data plan for free, in exchange for a commitment to use the phone as their personal device for six months or longer. This approach cannot solve the privacy issue. With such a usage model, the researchers from different research groups are not able to test their hypothesis at a world-wide scale or reuse each others' user base. For example, a researcher who uses PhoneLab at the University of Buffalo cannot share the same user base with Community Seismic Network [6] at Caltech, and vice versa.

Sensibility Testbed is different in several aspects. First, in our testbed, device owners participate as volunteers, and researchers request these devices through our server. This server, which is called a clearinghouse, mediates remote device access but does not store any personal data. As a result, Sensibility Testbed relieves researchers from recruiting participants and allows different groups to share all the devices used in the testbed. Additionally, Sensibility Testbed does not require researchers to write full-fledged Android apps to perform experiments. Instead, it provides an easy-to-use Python-like programming interface. Last but not least, using Sensibility Testbed, device owners do not need to trust the researchers who run code on their devices. As you will see later in this article, Sensibility Testbed provides a secure, sandboxed environment for anyone to run experiment code on Android devices. This can effectively prevent potential security and privacy breaches.

Yet Another Testbed!

The Sandbox

Researchers run experiments in Sensibility Testbed by writing code for a restricted, Python-based sandbox. This is the same security-reviewed Repy (Restricted Python) sandbox [2] used in the Seattle testbed (https://seattle.poly.edu/). This sandbox has been deployed on the Seattle testbed over the last six years. Our experience has shown that the risk of it being faulty is very low. The Repy sandbox is restricted in that its API limits what a sandboxed program can do: reading from and writing to the file system can only occur in a per-experiment directory; sending and receiving data via the network interface cannot exceed a configured rate; CPU, memory, and battery consumption cannot exceed a limit, etc. Therefore, the sandbox isolates the program from the rest of the device. More importantly, the sandbox allows us to interject code to implement privacy policies and control what happens with the data gathered on the device. You will read more on how to add privacy policies a bit later.

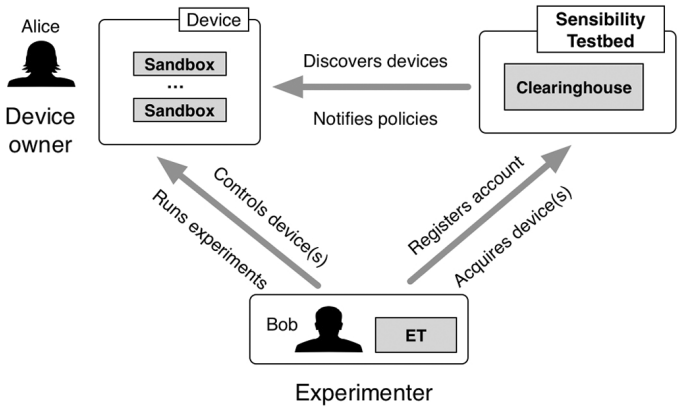


Figure 1: Sensibility Testbed architecture

Interacting Parties

In Sensibility Testbed, there are three categories of interacting parties: mobile devices owned by ordinary people, with our app installed; a clearinghouse server that discovers and configures participating devices; and researchers wanting to run experiments on mobile devices (see Figure 1). These three parties interact as follows. Mobile devices provide resources and data for researchers to use in their experiments. As mentioned above, researchers' code runs in a sandbox on a remote device that isolates the code from the rest of the device host system. Meanwhile, our clearinghouse server helps researchers acquire and manage devices, and enforces policies specified by the researcher's institutional review board (IRB), thus protecting device owners' personal information. Finally, researchers use their local machines to initiate and control experiments in Sensibility Testbed. They use an experiment tool (ET) to deploy and run experiments in sandboxes on remote devices that are acquired through the clearinghouse.

How Does It Work?

To get a sense of the technical details, let's walk through two scenarios: (1) a smartphone owner, Alice, participates in the testbed, and (2) a researcher, Bob, runs code on Sensibility Testbed using Alice's smartphone, among other devices. Specifically, Bob wants to know the cellular service quality in major cities. As such, he needs location information of individual devices, their cellular service provider, network type (3G, 4G, LTE, etc.), and signal strength. Note that the exact nature of Bob's experiment, be it collecting data, performing computation, etc., is not critical at this point due to code containment by our sandbox.

When Alice decides to participate in Sensibility Testbed, she first goes to the Google Play Store to download our Sensibility Testbed app [7]. The app contains sandboxes for researchers to run experiments on Alice's device, and a user interface for Alice to start and stop the app. When the app is started, Alice's device can be discovered by the clearinghouse. To keep track of Alice's device, the clearinghouse uses a database that stores her device's unique public cryptographic key that is generated during installation. This key is not associated with Alice's or her device's identity, but only the installation on the device. If Alice ever uninstalls the Sensibility app, this key is deleted, which effectively "unlinks" her device from any metadata stored on the clearinghouse. Instead of uninstalling, Alice may also choose to opt out of individual experiments.

To run code on Sensibility Testbed, Bob provides a detailed experiment description to our clearinghouse. Before Bob can request a device, his experiment needs to be approved for human-subjects compliance by his IRB (or equivalent). The IRB at Bob's institution specifies what data can be accessed by a research experiment, at which granularity or frequency of such data can be accessed, and so on. For example, Bob's experiment can (1) read location information from devices at the granularity of a city; (2) read accurate cellular signal strength and network type, but no information about cell IDs should be accessed; and (3) get location and cellular network data updates every ten minutes. Bob submits an appropriate experiment description for these requirements, which the clearinghouse codifies into policies that are later enforced on remote mobile devices.

Note that Bob cannot request access to all sensors at any rate even if his IRB approves such a policy. The Sensibility Testbed's IRB allows access to sensors in a way that is low risk, whose access can be pre-approved with the researcher's local IRB. However, we do not provide unfettered access to all sensors. Access to sensors of higher risk needs to go through the Sensibility Testbed's IRB, in addition to the researcher's IRB. However, for most cases, we expect that researchers need only go through their local IRB to get the sensor access they need for their experiments.

Bob next obtains an experiment account and requests a number of devices from our clearinghouse. The clearinghouse looks up available devices, finds Alice's phone is available (among others), assigns it to Bob's experiment account, and instructs the sandbox on her device to apply data access policies for Bob's experiment: for policy 1 above, the sandbox blurs the location information returned from Alice's phone down to the coordinates of the nearest city; for policy 2, the sandbox blocks the access to cell IDs; for policy 3, the sandbox limits the rate of GPS location and cellular network queries from Bob's experiment to one every ten minutes. Bob then uses the experiment tool (ET) on his local computer to access Alice's device and do experiments. After collecting the data he needs, Bob can either use ET to download data from the remote devices from time to time, set up his own server to store all the data, or use a data store service we provide (Sensevis: https://sensibilitytestbed.github.io/sensevis/).

If Bob stores data at his own server, he must use protective measures to ensure that the data sent from the mobile devices is properly encrypted and that the server storage cannot be tampered with by any other parties. For example, Bob needs to register his server by providing the server's certificate and URL to our clearinghouse. The clearinghouse then instructs the devices accessible to Bob that all the sensor data collected should be sent to this server. The sandboxes on these devices then issue HTTPS POST using the server's certificate, and send encrypted data to Bob's server. After the data is collected, how to store the data securely is mandated by Bob's IRB.

Implementing Policies

To understand how policies are implemented, we need to start with the Sensibility Testbed app. The app on Alice's device contains a native Android portion, and our Repy sandbox. When Alice starts the app, the native code initializes a Python interpreter, launches the Repy sandbox, and starts the communication between the device and our clearinghouse. The sandbox's restricted, secure API provides calls to file system, networking, threading functions, and so on. Therefore, Bob's code can read files, send data through the network, etc. from Alice's device. However, the original Repy sandbox does not include calls specific to mobile devices, such as GPS location, WiFi network, Bluetooth, accelerometer, cellular network, etc.

To obtain smartphone-specific data, we first implemented our sensor API using native code in the Sensibility Testbed Android app. The Repy sandbox then uses RPC to invoke the corresponding Android code, and returns the data from native code to a sandboxed program. The Repy sandbox thus defines the sensor API as a set of higher level calls, such as `get_location()`, `get_wifi()`, `get_accelerometer()`, and so on. Our Wiki page [8] hosts the current ever-growing list. As such, the original Repy interface and the added sensor API together provide the complete "OS level" sandbox kernel on a mobile device, as shown in Figure 2.

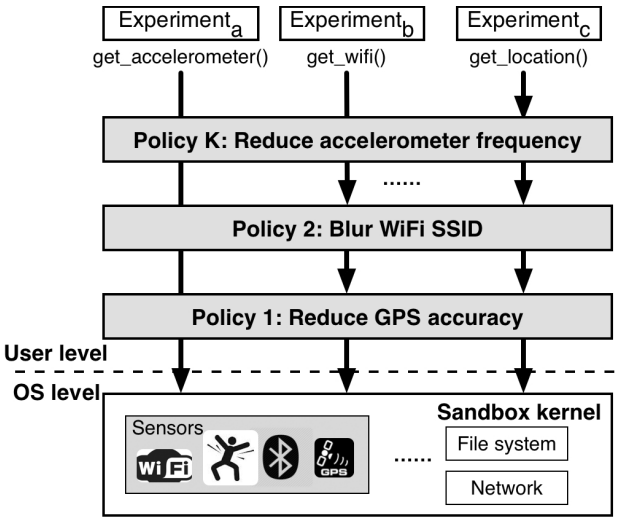


Figure 2: Sensibility Testbed blur policies

Finally, this sandbox kernel determines how policies are implemented by affecting API calls. It can interpose on a call and modify the data returned, or control how frequently a call can be made over time. As mentioned above, Bob provided his IRB policies through our clearinghouse. So before Bob runs his experiment, the clearinghouse instructs the sandbox on Alice's device to restrict sensor access in accordance with these IRB policies. Using the `get_location()` call as an example, when Bob's code requests location data from Alice's device, the Repy sandbox first invokes the location-related Android code. As the location data is returned, Bob's IRB policy indicates that the returned location coordinates should be blurred to the nearest city to Alice's device, instead of her actual location. As a result, the sandbox returns an approximate location to Bob's program. Furthermore, as Bob's IRB policy disallows collecting information about cell tower IDs, the access to cell IDs is blocked entirely on Alice's device. Similarly, other information like WiFi SSID can be blurred to a hashed string, the frequency to access an accelerometer can be restricted to prevent inferring passwords from the movement and tilt of the device, and so on. As shown in Figure 2, different policies can be stacked together as a set of filters for different sensors before a sandboxed program can access the sensor data.

Testbed Status

At this stage, multiple groups have experimented with Sensibility Testbed on their local phones, while we finalize outside use via our internal IRB and clearinghouse. We have also hosted two successful hack-a-thon-styled workshops with IEEE Sensors Applications Symposium [9] in 2014 and 2015. At these events, about two dozen participants from diverse universities and backgrounds worked in teams to build an application of their choice. Despite having no background in the platform and only a few hours to work, the participants in seven teams built applications that they demoed at the conference. The applications they developed varied from building automation using Bluetooth, to auto-device power saving that shuts down unnecessary network interfaces.

Sensibility Testbed provides a focal point for smartphone-based research. We believe this will bring benefit to researchers, as we make their experiment prototype faster, the remote control and management of devices easier, and running experiment code more secure. This will also benefit device owners in the long run, as researchers identify opportunities for improving current network protocols or systems, and implement or evaluate new services, algorithms, and research ideas. We believe Sensibility Testbed will bring more opportunities to research and encourage innovation from the general research community.

Sensibility Testbed is an open source research project, and we invite you to participate too! All of our source code, including the Android app, sandbox, clearinghouse, the experiment tool, etc., can be found on GitHub [10]. By installing the Sensibility Testbed application, you can become an important part of research discoveries that benefit science and technology.

References

[1] "A Warning About Those Free Smartphone Apps": http://abcnews.go.com/Technology/warning-free-smartphone-apps/story?id=30484903.

[2] J. Cappos, A. Dadgar, J. Rasley, J. Samuel, I. Beschastnikh, C. Barsan, A. Krishnamurthy, T. Anderson, "Retaining Sandbox Containment Despite Bugs in Privileged Memory-Safe Code," *ACM Conference on Computer and Communications Security (CCS '10)*.

[3] "App Auto-Tweets False Piracy Accusations": http://yro.slashdot.org/story/12/11/13/2249203/app-auto-tweets-false-piracy-accusations.

[4] "Hackers Are Draining Bank Accounts via the Starbucks App": http://money.cnn.com/2015/05/13/technology/hackers-starbucks-app/index.html.

[5] Sensibility Testbed: https://sensibilitytestbed.com/.

[6] Community Seismic Network: http://csn.caltech.edu/.

[7] Google Play Store, Sensibility Testbed: https://play.google.com/store/apps/details?id=com.sensibility_testbed.

[8] Sensibility Testbed, Using Sensors: https://sensibilitytestbed.com/projects/project/wiki/sensors.

[9] IEEE, Sensors Applications Symposium: http://sensorapps.org/.

[10] GitHub, Sensibility Testbed: https://github.com/SensibilityTestbed.