This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication.

Four Years Experience: Making Sensibility Testbed Work for SAS

Yanyan Zhuang
University of Colorado, Colorado Springs

Albert Rafetseder
New York University

Richard Weiss
Evergreen College

Justin Cappos
New York University

***Abstract*—Sensibility Testbed is a framework for developing sensor-based applications that can run on user-provided smartphones, and is easy to program. Over the past four years, we have been organizing hackathons at SAS in order to perform semi-controlled experiments with this platform. Any smartphone user can install Sensibility Testbed and develop a simple sensor application in less than a day. One of the problems with developing and testing such a framework is that there are many possible hardware platforms and system configurations. Hackathons provide an effective venue for observing the development of applications on a range of devices by users with no previous knowledge of the framework.**

In this paper, we describe our experiences with hosting hackathons in a variety of venues, including the challenges of working in unfamiliar environments and with researchers who had no prior knowledge of the testbed. The feedback from participants has been very useful in identifying usability issues, hardware issues, and the types of sensor applications that users want to create.

I. Introduction

End-user mobile devices, such as smartphones and tablets, have become indispensable gadgets in people's everyday lives. Research has shown that nearly two-thirds of Americans own a smartphone, and 19% of them use the phone as their only means of staying connected. For many people, these devices have become the dominant way they interact with others, and with the physical world.

Given the sheer number of these devices and the increasing sophistication of their sensors[1], the value of smart devices as data collection vehicles for government, university, or corporate studies continues to grow. Since these devices have GPS, accelerometers, cameras, and microphones, they can generate valuable data for such studies as determining noise levels within an urban neighborhood [1], detecting approaching earthquakes [2], or studying traffic patterns at intersections that could be critical in an emergency [3], [4]. Accessing devices in a home network can help providers improve the quality of service [5]. Testing applications on remote devices also allows developers to better understand how applications perform in diverse environments, thus enabling improvements in performance [6]. For instance, some platform APIs change their behavior depending on the battery level of the device [7]. Without remote access to battery life, these APIs can not guarantee basic function efficiency. As a result, a number of initiatives have been launched within the network community to study mobile devices (e.g., Mobilyzer [8]), and in the systems community to deploy new services and test research prototypes (e.g., Phonelab [9]).

In this work, we introduce Sensibility Testbed [10], [11], a mobile testbed that that runs on user-provided smartphones and is easy to program. First, it has a secure sandbox that provides a programming language interface equipped with system calls for networking, file system, threading, locking, logging, and most importantly, sensors. Every system call is strictly sanitized to preserve consistent behavior across different OSes, and to avoid exploitable vulnerabilities. Second, it provides infrastructure and services that makes it easy for researchers to conduct an experiment. For example, a lookup service lets a researcher look for a device using a unique ID, even if the device changes its IP address. Additionally, our programming interface hides all the unnecessary details in mobile programming, thus a researcher only needs to write as little as one line of code per sensor to collect data.

In this paper, we will first introduce the design of Sensibility Testbed. In the past four years, we used Sensibility Testbed to host Sensor Application Development Workshop, co-located with Sensors Applications Symposium (SAS). Overall, we hosted four hackathon-styled workshops, and one localization challenge. We will thus describe our experiences hosting these events using Sensibility Testbed, what worked, and what did not work for us.

The rest of the paper is organized as follows. In Section II we present the design and implementation of Sensibility Testbed and its components. Section III describes the four workshops and a challenge we have hosted in the past four years. In Section IV we discuss our experience, what have worked and what have not. Section V gives an overview of related work, and Section VI provides our concluding remarks.

II. Sensibility Testbed Design and Implementation

This section describes the implementation of the techniques in Sensibility Testbed. Our testbed benefitted from the design and implementation of our prior experimental platform for networking and distributed system research called Seattle [12]. Deployed seven years ago, Seattle continues to run in a safe and contained manner on tens of thousands of computers around the world. A core design principle of its operation is the installation of secure sandboxes on end-user computers. These sandboxes limit the consumption of resources, such as CPU, memory, storage space, and network bandwidth, allowing

[1]In this work, we broadly define sensors as the hardware components that can record phenomena about the physical world, such as the WiFi/cellular network, GPS location, movement acceleration, etc.

978-1-5386-2092-2/18/$31.00 ©2018 IEEE

Seattle to run with minimal impact on system security and performance. By operating inside a sandbox, it also ensures that other files and programs on the computer are kept private and safe.

We first provide a high-level walkthrough of Sensibility Testbed, when a researcher conducts an experiment on a mobile device (Section II-A), and then present the different components of Sensibility Testbed (Section II-B).

A. Overview

Let's assume that device owner Alice participates in a Sensibility Testbed experiment, while a researcher Rhonda wants to run code on Sensibility Testbed using a number of devices, including Alice's.

To run code on Alice's device, Rhonda only needs to download an experiment manager (Section II-B3) to her own computer. The experiment manager is a light-weight command line console that can directly access Alice's device, upload experiment code, and communicate with Alice's device to start or stop the execution of the experiment. The experiment manager can also be used to download data from remote devices to Rhonda's local computer, or to a server she has set up to store the data.

B. Sensibility Testbed Components

1) Secure Sandbox: The sandbox in Sensibility Testbed provides a programming language interface equipped with system calls for networking, file system access, threading, locking, logging, and most importantly, sensors. Every system call in the sandbox is strictly sanitized to preserve consistent behavior across different OSes, and to avoid exploitable vulnerabilities. Additionally, the sandbox interposes on system calls that use resources, such as network and disk I/O, and prevents or delays the execution of these calls if they exceed their configured quota [13]. Therefore, different researchers can run experiments on different sandboxes on the same device, without any interference between the experiments or the rest of a device. Most importantly, the same system call interposition technique can modify a system call's behavior, such as the return value of a call and the frequency.

2) Device Manager: The Sensibility Testbed device manager is part of the Sensibility Testbed app that device owners install on their devices. It allows device owners to control the experiments running on their devices. The Sensibility Testbed app starts as the device boots, and runs as a background process. The device manager also includes code to dynamically traverse NAT gateways and firewalls (both are commonly found on Internet connections over WiFi), so that it remains contactable even if the device does not currently possess a public IP address.

3) Experiment Manager: Researchers use an experiment manager to directly access remote devices, much like using a command line console via `ssh`. The experiment manager, including assorted sandbox libraries to support the development of experiments, is available in packaged form for downloading from our project website (the light-weight command line console in Section II-A). It runs on different operating systems and platforms. To accommodate the complexity of today's networks, the experiment manager supports contacting devices in WiFi networks, and networks behind NAT gateways and firewalls.

4) Clearinghouse: Sensibility Testbed's clearinghouse is hosted on a web server at New York University. It is implemented based on the Django web framework. The clearinghouse stores information about available devices and registered researcher's experiments. Furthermore, it possesses cryptographic keys that allow it to assign a researcher access to sandboxes. Similarly, when the researcher's experiment period is over, the clearinghouse removes her access rights from every sandbox she previously controlled.

III. SAS Workshops

We have been organizing and hosting workshops with SAS since 2014. This has benefitted the project in two ways: it contributes to testing and it suggests new features and improvements to the platform. In addition, we developed a sensor app for indoor localization. Sensibility Testbed itself also evolved over the past four years. Notably, it evolved from a simple Android app based on an XML-RPC [14] interface, to developing our own embedded Python interpreter in C that can call into a running JVM to read out sensor values. In this section, we look back on our experience organizing four hackathons and one localization challenge, as well as the evolution of Sensibility Testbed.

A. Hackathons

1) 2014 — Queenstown, New Zealand: This was the first year we hosted a hackathon using Sensibility Testbed. The system was in a very early stage of its development, and the design and implementation was far from mature. However, this first year was a success despite all the challenges.

The first version of the Sensibility Testbed app depended on an XML-RPC interface. Since Sensibility Testbed supports Android, the app was developed in Java. However, Sensibility Testbed's secure sandbox was written in a programming language similar to Python, and thus the experiment a researcher developed must be written in a way similar to Python. In order for the Python code to access sensor data provided by Android Java interfaces, we let the native Android app and the Python sandbox run at the same time. The Android app opens up a port for communication, providing interfaces for sensor data whenever the data is available. If an experiment (written in Python) establishes a connection with the Android app using this designated port, then the experiment can invoke the Android app's Java interface, using a remote procedure call, to obtain sensor data. To enable all the sensor interface in Java, we relied on a third-party library called Scripting Language for Android (`sl4a`).

We spent about an hour at the workshop presenting our system to the participants, and teaching them how to use Sensibility Testbed to write programs that read sensor data. Then the participants spent the rest of the day working in

groups, using Sensibility Testbed to develop an app. At the end of the day, each group pitched their work at the conference banquet, where we asked the conference organizers to judge their work. Certificates were given to the top three teams. Team members on the first place team each received a new Android phone.

In the first hackathon, we had about 20 participants. The top team was from the University of Houston. They built an app that monitors the battery level of a device, and whether WiFi and Bluetooth were turned on. If battery is low, the app uses text to speech to tell the device user to turn off the WiFi or Bluetooth. Other teams built exciting apps as well, e.g., using accelerometer, GPS, and noise data.

Takeaway: The first year was critical for us. We learned that simple installation and good documentation are the most important elements of a successful hackathon. We carried on these important elements in the following years.

2) 2015 — Zadar, Croatia: As the first hackathon was well-received, we were invited to host another one in the following year in Croatia. With the experience we had in the first year, we followed the same agenda. The second hackathon was also successful.

However, during this second year, we discovered some weaknesses of our system. For example, the XML-RPC interface slowed the access frequency for the accelerometers and gyroscope from 150 Hz to 50 Hz. This slowdown was noticeable and introduced inaccuracies in some experiments. Furthermore, XML-RPC is not a secure communication channel. If an attacker learns about the designated port, he or she can use an XML-RPC call to get all the sensor data as is desired.

Another issue was caused by `sl4a`. Originally, the Sensibility Testbed app automatically downloaded the `apk` for `sl4a`, and installed it within the Sensibility Testbed app. However, in early 2015, Google Play Store disallowed this in-app installation behavior. We then changed our app such that the app would prompt the user to install `sl4a` manually. As a result, the user had to allow "installing apps from unknown sources" in the device settings, as `sl4a` was not an app in the Play Store. This added an additional burden to the participants of our hackathon. Therefore, we planned for some significant changes in the next year.

Takeaway: The unexpected findings helped us discover the problems, and solve them in a timely manner. For the XML-RPC issue, we needed a new way for experiment code to get sensor data. To resolve the issue related to `sl4a`, we needed to develop our own sensor interfaces.

3) 2016 — Catania, Italy: The changes we made in 2016 were fundamental, as we got rid of the XML-RPC interface and `sl4a` entirely, from the lessons we learned from the previous year. In order to improve the efficiency and security of the data access, we re-engineered the interface between Sensibility Testbed's Python sandbox and the Android operating system.

In this new design, we interfaced the native Sensibility Testbed Android app with the Python-based sandbox. This is achieved by using the Java Native Interface (JNI) to define

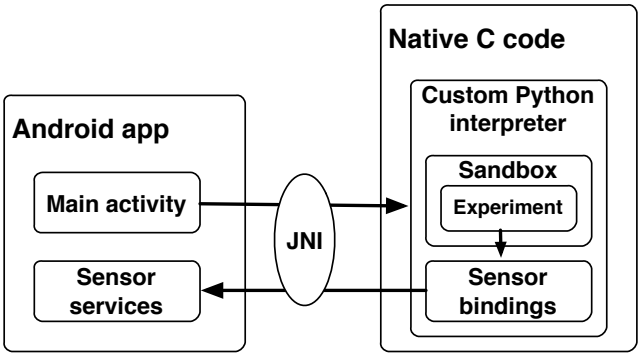

Fig. 1. Sensibility Testbed Android app and Python sandbox architecture.

interfaces into the Android app on one side, and a custom Python interpreter that then hosts the Sensibility Testbed sandbox on the other side. Custom sensor bindings were written and compiled into the Python interpreter using the CPython API. That way, Android sensors became callable and readable from within the Python interpreter in the form of built-in modules. On top of that, we created wrapper functions for the sandboxed code so that the extent of sensor access can be tailored. We can therefore implement policies to protect a user's privacy. Figure 1 shows this architecture.

With all these modifications, we were able to completely discard `sl4a`, as well as the slow and unsafe XML-RPC interface. However, the support efforts on the development end increased as a result of this design decision. This is due to our use of the Android Native Development Kit (NDK) in the creation of the augmented Python interpreter. The NDK is a low-level toolkit for compiling C and C++ code for Android (as opposed to the higher-level Java and Kotlin toolkits available, which account for virtually all of the code written for the platform.) Since the NDK sees less use, its version changes frequently introduced "breaking changes", and the Sensibility Testbed development team had to track updates carefully.

However, the end result for Sensibility Testbed users is positive. In this year's hackathon, participants were able to install only one app, instead of two, and use an experiment manager to easily interact with a device and get sensor data at almost native rate. For example, if a device's rate is 150 Hz, the experiment through our sandbox read the sensor at almost 150 Hz. As a result, it was much easier for the participants to get an experiment running, and get the data they wanted. A group with members from Qatar and Norway developed an app that detected the slips and fall of a device owner, and report the event using text-to-speech.

Despite the progress, we encountered some unexpected difficulties this year. As the conference was hosted in a museum, the WiFi connectivity was very sporadic, and the quality was poor. Participants had difficulty connecting to their devices while conducting their experiment, or the connection

would break from time to time. Eventually, we were able to borrow a WiFi router from the conference and set it up for the hackathon.

Takeaway: Despite the technical progress, non-technical issues can still be our obstacles. We need to prepare fully in all aspects.

4) 2017 — Glassboro, New Jersey, USA: With the lesson we had in 2016, we devised new strategy: we brought our own WiFi router and setup a hotspot in the conference room where we hosted the hackathon. We also brought our technical team to troubleshoot any issues on the spot. Last but not least, we improved our documentation about Sensibility Testbed to a significant degree, where it is crystal clear what steps should be followed.

With all previous years' effort and our new changes, 2017 was the year we had the least number of issues. A group with members from Italy and Canada developed an app that by scanning the WiFi network at the conference, it recommended which WiFi router had the best connectivity quality in terms of received signal strength.

Takeaway: Hard work pays off. We will continue our effort in 2018.

B. Localization Challenge

Some of the work to improve sensing rates was motivated by our own internal development of a sensor app for indoor localization. The app was designed to use the accelerometer to count steps and the gyroscope to measure the heading of the device.

In 2016, we hosted a localization challenge using Sensibility Testbed. The conference was held in a museum, a venue with a relatively complex floor plan. Our team concluded that this would be a good test of the accuracy and usability of the app as well as an opportunity to collect accelerometer data for testing localization algorithms, in general.

The indoor localization challenge is the problem of determining where a device is located in an indoor environment where there is no access to GPS. It has many possible applications, from allowing a user to navigate to a room, to tagging video streams with camera location. The first issue with this problem is defining what is meant by location. Without GPS, there is no natural reference coordinate system. For our app, we create fiducial or reference points that define a coordinate system, and all locations are computed with respect to those coordinates. For our challenge, we had no access to building blueprints or any prior information about the venue, so we needed a system that was easy to set up and reliable. Although it would have been possible to use WiFi or Bluetooth beacons, we chose an approach that required less infrastructure. The location markers were QR codes printed on paper and taped to the walls.

Using QR codes to create reference points worked well. The locations of the QR codes with respect to each other were easily measured using a laser range-finder. The only constraint was that we needed to arrange the points in triangles where each vertex was visible by the other two. As long as we could decompose the space into triangles such that each triangle shared an edge with another triangle, the coordinate system was well-defined. Although the room was not rectangular, and there were alcoves, we were able to find triangles that covered the space and set up the reference points.

Users of the indoor localization app were requested to walk around the space and occasionally capture one of the QR codes using their phone. This allowed the app to compute the location and orientation of the phone at specific times. In between these measurements, the app updated its position and orientation based on accelerometer and gyroscope measurements, integrated through a Kalman filter.

One of the issues that we encountered was that some attendees had iPhones, and our app only works on Android. Even for Android, the app did not seem to install on all of the older versions. In those cases, we were able to lend the participants one of the mobile phones that we brought. Another issue was related to the lack of WiFi connectivity in the room where we conducted the experiments. Our app was designed with the assumption that mobile phones could connect frequently to the Internet and upload data to a server that we had set up. This was a problem because the app was collecting large amounts of data, some of which was lost when the phone ran out of disk space. However, the experiment was successful and we collect a number of traces with accelerometer and gyroscope data together with reference points. The data sets can be found in our repository.

Takeaway: App developers cannot assume that user devices will have frequent Internet connectivity.

Similar work had been done by Microsoft. The Microsoft Indoor Localization Challenge was held at IPSN in 2014 [15]. At that event, there were two categories: structured and structureless, and teams collected their own data. Infrastructure-free approaches used existing WiFi and geo-magnetic signals. Infrastructure-based approaches relied on the deployment of customized RF-beacons, infrared, ultrasound, and Bluetooth. The study concluded that indoor localization with 1-meter accuracy was still an unsolved problem.

IV. Discussion and Lessons Learned

Due to the many possible hardware platforms and system configurations brought by the participants, our platform has to keep up with the new features and environments (Section IV-A) while still being easy to use (Section IV-B). Furthermore, it has to balance security with usability (Section IV-C). In this section, we describe what we have learned by hosting hackathons with Sensibility Testbed.

A. Working in Fast-changing and Unfamiliar Environments

1) Smartphone Device: In principle, Android is based on the open-source Linux kernel; however, it is a fast-changing operating system that keeps incorporating nonstandard additions and changes which break one-to-one Linux compatibility for our sandbox. In the past, Android updates have caused a number of problems. For example, since Android 5.0, Google included a number of system behavior changes such

as forbidding executable files in the user-accessible parts of the device's file system. These and other changes required substantial changes to the internal layout of our app, and also to the core parts of the sandbox.

Furthermore, our Sensibility Testbed app currently supports ARM CPU with a 32-bit instruction set. As ARM/64-bit becomes more popular, we will need to support that, as well as other ISAs.

2) Participants' Laptops: From the past four hackathons, we found that the majority of our participants had Windows laptops, which do not have Python installed. In contrast, Python is preinstalled on most Linux distributions and Mac, and is available as a package on all others. Therefore, Windows users had to install Python manually, with the right version 2.7. As a result, we had to provide extra guidance to participants that they install the correct Python environment.

3) Network Connectivity: Since a participant establishes an `ssh`-like connection with a smartphone device, a bad network connection can frustrate the participant after both the smartphone and the laptop have been set up. Furthermore, if the participant changes location and thus changes the device's IP address, it will take a while before the new IP address can be recognized by the participant's laptop. For this issue, we added buttons in the app to refresh the state of a device, or instructed the participant to restart the connection on the laptop end. However, this is not a sustainable and long-term solution.

Lessons Learned: As the environment is crucial for running experiments, we will need to keep up with the new features of Android, and any potential new environments that we may want to target, such as iOS. On the other hand, providing Windows users with more detailed instructions can lead to a more smooth user experience. For network connectivity, we currently plan to work on a version of the laptop-side software so that the laptop and smartphone can work in tethering mode. To sum up, we need to work out or work around issues as they emerge, while keep the system user-friendly.

B. Working with Researchers with No Prior Knowledge

1) Command Line: Many researchers have not worked with a command-line console before our hackathon. Therefore, in many cases we had to teach participants how to navigate file systems, run commands, etc.

2) Lack of familiarity with Python or Sensibility Testbed: Some participants had never programmed in Python. But fortunately, the Sensibility Testbed API (based on Python) is very easy to understand and use. We provide documentation for each API call's syntax, as well as intuitive examples. We found that only occasionally we had to explain the APIs to the participants or provide hands-on assistance.

Although none of the participants had any prior experience with Sensibility Testbed, we found that by following our detailed tutorials, most of the participants were able to write experiments in a very short amount of time. Among the 25 teams that have participated so far, only one group did not finish the application development.

Lessons Learned: Despite lack of experience from the participants, our documentation effort has played a critical role in the past hackathons. Although it is very difficult to document a large and complex system, this work is very important for the usability of the system.

C. Usability vs. Security

As Sensibility Testbed is a testbed running on volunteers' devices, security and privacy is a primary goal (Section II). However, in order to make strong claims of privacy protection, it was necessary to relax some of the usability features, and we had to strike a balance between usability, security and privacy. For example, in Sensibility Testbed, the programming interface disables some sensor access that are a risk to privacy, such as cameras and microphones [11]. During the hackathons, quite a few participants requested access to these sensors, for applications like facial recognition and intrusion detection. However, due to the high risk, we decided not to enable access to these sensors. Although this made some application impossible to implement, the resulting security benefit was greater.

We also made improvements once we identified initial design flaws, e.g., `sl4a` both added burden to participants to install a separate app, and also was not secure due to the XML-RPC interface (Section III-A). By removing this component, both the usability and security aspects of Sensibility Testbed were greatly improved. Although the effort was significant, our experience from the hackathons provided a strong incentive to make this change.

Lessons Learned: It is difficult to achieve both usability and security, therefore we need to make tradeoffs in our design and implementation. As we gain more momentum in Sensibility Testbed's use, and get more feedback from participants, we will be able to find out which usability/security features worked and which did not. Additionally, the Android operating system has made its security model stronger over time, such as requesting permissions from the user. Since our security model is built on top of that, we need to keep up with Android's security model.

V. Related work

As a smartphone testbed, Sensibility Testbed tries to balance the security and usability. Our testbed also draws upon the experiences of other platforms, though most of the other platforms do not take usability into consideration.

PhoneLab [9] is a smartphone testbed that is the most close to our testbed. However, there are significant differences that make Sensibility Testbed more suitable for a conference like SAS. First, researchers using PhoneLab have to write their own experiment in Android native code, and submit the experiment in the Google Play Store. This process alone can take significant time. Second, the data collected from PhoneLab devices is sent to an external server. Researchers can only get the data when an experiment is over. Finally, researchers do not have the permission to decide when to run an experiment, or which devices to choose from. Such

scheduling is done by the PhoneLab services. Therefore, from the researcher's point of view, PhoneLab is not flexible enough to learn within a day.

There are also other testbeds for the mobile and wireless network, or sensor network community. For example, MobiLab [16] is a wireless sensor network testbed for carrying out repeated and reproducible experiments. It is a controlled network testbed that can ensure experiments produce predictable outcomes, despite the high fluctuation of link quality during mobility. In contrast, Sensibility Testbed embraces the unpredictability of wireless experiments which is a more realistic environment.

The work of Zhao, etc. [17] presented a federated semantic for Internet of Things (IoT) testbeds. This allows highly heterogeneous IoT experiments to be interoperable, instead of being fragmented or silo solutions. However, this type of testbeds rely on existing experiments, and cannot stand on their own and serve as a platform for researchers to freely conduct research experiments.

OneLab [18] has been used at TridentCom. However, OneLab is also a testbed for federating various testbeds, such as PlanetLab Europe (PLE), an IoT testbed, a cognitive radio testbed, and a wireless testbed. Even if OneLab portal is easy to use, deploying experiments across federated platforms still requires researchers to know how to work with each individual testbed. Therefore, not only the platform cannot stand on its own, the learning curve is overall higher than Sensibility Testbed.

VI. Conclusion

We described our past four year's experience using Sensibility Testbed to host hackathons with SAS, by briefly introducing the system, the effort we made to host hackathons and localization challenge successfully, the problems we ran into, and the lessons learned. In the design of Sensibility Testbed, we struck a balance between usability and security. At the SAS conference, we work with unfamiliar environments and with researchers who had no prior exposure. Sensibility Testbed thus has evolved as we gain experiences each year, and has become more user-friendly while secure at the same time. All these experiences will help us improve Sensibility Testbed further, make it easier to use, and host more successful hackathons with SAS in the future.

References

[1] C. A. Kardous and P. B. Shaw, "Evaluation of smartphone sound measurement applicationsa)," *The Journal of the Acoustical Society of America*, vol. 135, no. 4, pp. EL186–EL192, 2014.

[2] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, and A. Krause, "The next big one: Detecting earthquakes and other rare events from community-based sensors," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 13–24.

[3] Y. Zhuang, J. Pan, Y. Luo, and L. Cai, "Time and location-critical emergency message dissemination for vehicular ad-hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 1, pp. 187–196, 2011.

[4] Y. Zhuang, J. Pan, and L. Cai, "A probabilistic model for message propagation in two-dimensional vehicular ad-hoc networks," in *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. ACM, 2010, pp. 31–40.

[5] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 134–145.

[6] L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajan, I. Obermiller, and S. Shayandeh, "Appinsight: Mobile app performance monitoring in the wild." in *OSDI*, vol. 12, 2012, pp. 107–120.

[7] J. Spooren, D. Preuveneers, and W. Joosen, "Leveraging battery usage from mobile devices for active authentication," *Mobile Information Systems*, vol. 2017, 2017.

[8] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao, "Mobilyzer: An open platform for controllable mobile network measurements," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 389–404.

[9] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen, "Phonelab: A large programmable smartphone testbed," in *Proceedings of First International Workshop on Sensing and Big Data Mining*. ACM, 2013, pp. 1–6.

[10] Y. Zhuang, L. Law, A. Rafetseder, L. Wang, I. Beschastnikh, and J. Cappos, "Sensibility testbed: An internet-wide cloud platform for programmable exploration of mobile devices," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 139–140.

[11] Y. Zhuang, A. Rafetseder, Y. Hu, Y. Tian, and J. Cappos, "Sensibility testbed: Automated IRB policy enforcement in mobile research apps," in *19th International Workshop on Mobile Computing Systems and Applications (HotMobile18), Tempe, Arizona, USA*. ACM, 2018.

[12] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," *ACM SIGCSE Bulletin*, vol. 41, no. 1, pp. 111–115, 2009.

[13] J. Cappos, A. Dadgar, J. Rasley, J. Samuel, I. Beschastnikh, C. Barsan, A. Krishnamurthy, and T. Anderson, "Retaining sandbox containment despite bugs in privileged memory-safe code," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 212–223.

[14] A. Rafetseder, F. Metzger, L. Pühringer, K. Tutschku, Y. Zhuang, and J. Cappos, "Sensorium-a generic sensor framework." *Praxis der Informationsverarbeitung und Kommunikation*, vol. 36, no. 1, p. 46, 2013.

[15] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen, "A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, ser. IPSN '15. New York, NY, USA: ACM, 2015, pp. 178–189. [Online]. Available: http://doi.acm.org/10.1145/2737095.2737726

[16] J. Wen, Z. Ansar, and W. Dargie, "Mobilab: A testbed for evaluating mobility management protocols in wsn," in *Testbeds and Research Infrastructures for the Development of Networks and Communities*. Springer, 2016, pp. 49–58.

[17] M. Zhao, N. Kefalakis, P. Grace, J. Soldatos, F. Le-Gall, and P. Cousin, "Towards an interoperability certification method for semantic federated experimental iot testbeds," in *Testbeds and Research Infrastructures for the Development of Networks and Communities*. Springer, 2016, pp. 103–113.

[18] L. Baron, R. Klacza, N. Kurose, M. Y. Rahman, C. Scognamiglio, T. Friedman, S. Fdida, and F. Saint-Marcel, "Onelab tutorial: A single portal to heterogeneous testbeds," in *TridentCom 2015*, 2015.