EavesDroid: Keystroke recovery using mobile phone accelerometers

Akshay Mittal
Princeton University

Wathsala Vithanage
Princeton University

Stephen Lin
Princeton University

Jennifer Guo
Princeton University

ABSTRACT

[to - fill]

1. INTRODUCTION

Mobile phones [1] are becoming increasingly powerful devices. In addition to being able to run applications ranging from email clients to web browsers, a progressively sophisticated set of sensors are enabling these devices to more actively interface with the world around them. From gestures captured by accelerometers for games to augmented reality applications displaying metadata tags on video from the camera in real-time, mobile phones are becoming adept at capturing and harnessing rich features and data from their surroundings.

Unfortunately, the array of sensors in these devices can also be used in unintended ways. As many have suggested, malware could potentially gain access to a mobile phone's camera to take photos or video of the owner and their surroundings [2]. In cases where the camera may not be pointed at an interesting target, a malicious application could instead attempt to activate the device's microphone and record ambient sounds or syphon GPS data to track the target's location [3, 4, 5, 6]. Recognizing the potential for the leakage of sensitive information, many modern mobile phone operating systems provide mechanisms by which access to such sensors is protected by explicit permission from the user. For instance, the manifest files included with every Android application provide an unambiguous list of all of permissions an application may ever request at installation time. Should an application not request access to these resources or the user deny the application such rights, the application will not be able to potentially abuse these resources. However, access to all of the sensors contained in these devices are not so tightly controlled.

In this paper, we demonstrate that unfettered access to the accelerometers available on many mobile phones can allow for significant unintended leakage of information from a user's environment. We show that a malicious application with access to the accelerometer feed can record and reconstruct the keypresses made on a nearby keyboard based solely on the observed vibrations. We develop profiles for keypress events using boosted Random Forests, which creates an abstract representation of the relationship between the keystroke signal and the alphabet typed. We then recover the typed content by translating from our intermediary form to English words using a number of different dictionaries. Such tasks are not a trivial application of standard techniques, especially when compared to previous efforts in this space. Specifically, we must overcome much lower sensor sampling rates than has been experienced in the related acoustic and electromagnetic-based eavesdropping attacks, which makes deciphering individual keypresses extremely difficult.

Through this, we make the following contributions in this paper:

1. **Develop an infrastructure for characterizing keypress vibrations**: We capture, analyze and develop profiles of keypress events on a nearby keyboard based on the vibrations created when they are pressed. Our inputs are then processed using a set boosted Random Forests to create an intermediary representation, which is combined with candidate dictionaries to successfully recover words at rates as high as __

2. **Dataset made publicly available**: To the best of our knowledge, there does not exists a dataset with the keystrokes for the vibrations recorded using an accelerometer. In this paper, we demonstrate we construct the dataset of keystroke vibrations, recorded using the accelerometer, present in a smartphone and make the dataset publicly available to enhance further research on this topic. Given the widespread about security breaches and email leaks, we perceive this to be of great importance in motivating the mobile operating system manufacturers to restrict the usage of accelerometers to permitted applications only.

Note that our approach is greatly influenced by the work of

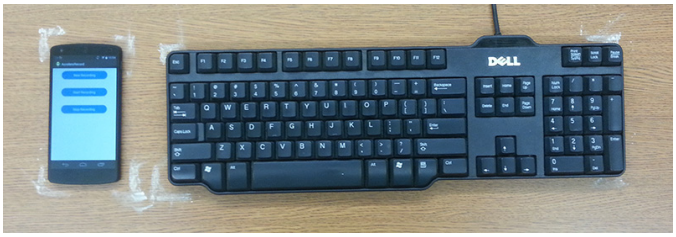

Figure 1: Threat model of a smartphone placed next to a keyboard. Used in our experimental setup to capture the dataset.

Marquardt *et. al* [1] and we present key differences with our approach.

2. RELATED WORK

Researchers have studied malicious code on mobile devices that learn information about the device owner using other embedded sensors. In the most closely related work to our paper, Marquardt *et. al* [1] propose *(sp)iPhone* which uses motion sensors in the iPhone 4 to infer keystrokes of nearby laptop users. *(sp)iPhone* uses 3 labels for each pair of keystrokes in order to determine the correct identification for the letters of the word. The location of the key is modelled using a neural network trained on 150 keystrokes for each letter of the English alphabet. A profile is then constructed for each pair using the neural network and determines and models the distance between the two events. The predictions of the neural network are matched against a dictionary to determine the top matches. The accuracy achieved is roughly 80% but this decreases to 40% as the size of the dictionary increases.

In a very similar, but orthogonal, use case, Owusu *et. al* [7] present the threat of applications which extract the keystrokes of the user's typing on the smartphone screen itself. Again, due to no restrictions on the usage of the accelerometer, they are able to extract 6-character passwords in as few as 4.5 trials(median). TouchLogger [8] uses orientation of the smartphone device to infer the keystrokes.

3. THREAT MODEL

Our attack is based upon two observations:

Many users like to place their smartphone close to their laptop while working, so that it is easily accessible and notifications are not missed.

Furthermore we note that while smartphones offer a broad range of sensors such as microphone, GPS, gyroscope and accelerometer, the accelerometer is the only sensor to which access is not protected in any current mobile phone operating system. That is mainly due to the frequent usage needs of the accelerometer to determine window orientation and update screen orientation, a feature any app can make use of.

4. FRAMEWORK DESCRIPTION

4.1 Dataset Capturing

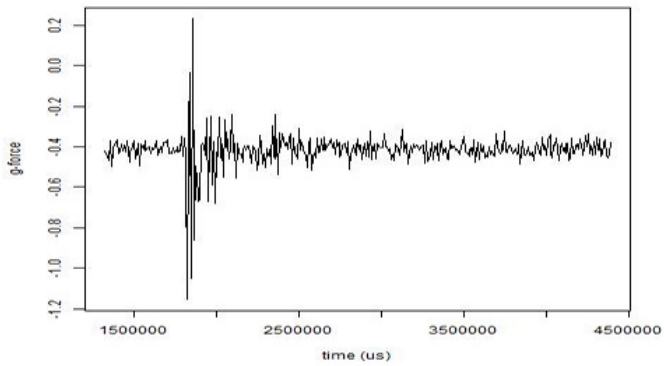

Figure 2: Sample Recording for letter 'a'

Our experimental setup is shown in fig.1. We placed a Dell USB keyboard and a LG Nexus smartphone on a wooden table. No other items were placed on or were touching the table. All keys were pressed with the index finger of the right hand. The hands do not touch the keyboard or table, and the fingers do not pause on the keyboard but only touch it during the brief duration of the key press.

We recorded 1000 datapoints in 40 sessions with 25 letters in each sessions. For each session we generate a sequence of 25 random letters and display the letters in 3 second intervals to ensure an even recording of the letters and to ensure that there is no bias towards a specific letter. For the data capturing we developed an Android app that reads from the hardware accelerometer and syncs with our laptops, where we further process the data. The raw accelerometer data as returned by Android contains the x, y and z-direction components of the acceleration of the vibration received. We calculate the G-force magnitude as $\sqrt{x^2 + y^2 + z^2} - g$ where $g = 9.81$ is the acceleration due to gravity.

The original recording will contain a large spike at the beginning and end due to the pressing of "Start" and "Stop" on the recording app. We cut off these noise signals and further segment the data into 25 individual data points. An example of a plot of one such signal is shown in fig.2.

4.2 Feature Extraction

We use a combination of time-domain and frequency-domain features to construct our feature vector.

5. IMPLEMENTATION

[table with all combinations of accuracies]

LR predictor and Up down predictor .

Adaboost with Random Forest

Mention triads did not give too good accuracies

6. EXPERIMENTAL RESULTS

[Box sentence: exact matches and possible choices for non-exact matches]

[Negative analysis, word watch, many potential matches when dictionary size increases]

7. CONCLUSIONS

7.1 Future Work

Mobile browsers having allowing access to accelerometer data,

http://blogs.adobe.com/cantrell/archives/2012/03/accessing-the-accelerometer-and-gyroscope-in-javascript.html

http://isthisanearthquake.com/

http://isthisanearthquake.com/seismograph.html

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011.

[2] Zhu Cheng. Mobile malware: Threats and prevention. *McAfee Avert*, 2007.

[3] David Dagon, Tom Martin, and Thad Starner. Mobile phones as computing devices: The viruses are coming! *Pervasive Computing, IEEE*, 2004.

[4] Liang Cai, Sridhar Machiraju, and Hao Chen. Defending against sensor-sniffing attacks on mobile phones. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 31–36. ACM, 2009.

[5] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, 2010.

[6] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. Pios: Detecting privacy leaks in ios applications. In *NDSS*, 2011.

[7] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012.

[8] Liang Cai and Hao Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security*. USENIX Association, 2011.

APPENDIX

A. APPENDIX SECTION

A.1 Appendix Subsection