Detecting Phone Theft Using Machine Learning

Author Names Omitted for Anonymous Review. Paper-ID XXX

ABSTRACT

Millions of smartphones in the U.S. are stolen every year, exposing victims' private information. To protect smartphone users, we developed a system to automatically detect grab-and-run theft. We use binary classifiers to classify theft and normal usage with features extracted from accelerometer data. We collected a data set from simulated theft experiments and a user study of normal smartphone usage and built three models, among which random forest had the best performance. Random forest classifier could detect 95% of theft at no cost of false alarms.

CCS CONCEPTS

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

KEYWORDS

Phone-Theft Detection, Machine Learning, Usable Security

1 INTRODUCTION

According to the Consumer Reports, 2.1 million smartphones were stolen in the United States in 2014, which consisted of 1.5% of all smartphones[1]. The Pew Internet Project reported in 2012 that nearly one third of mobile phone users have experienced a lost or stolen device [2]. Lookout estimates that phone lost could cost U.S. consumers $30 billion per year [3]. In addition, hardware loss is not the only risk of phone theft. Egelman et al. indicates that 42% of users do not lock their smartphones, and this allows thieves to gain access to their personal information stored on their phones [4].

In this paper, we present a novel method of applying machine learning algorithms with accelerometer data to automatically detect grab-and-run smartphone theft. We build binary classifiers to classify theft and normal usage. A positive dataset was collected through designed experiments that simulated three typical theft scenarios. A negative dataset was obtained from a user study, where we collected sensor data, including accelerometer, from 55 participants' smartphones in 3 weeks. In order to generate training and validation datasets, we selected 7 features and featurized the magnitude of accelerometer data in one-second windows before and after it exceeded $40m/s^2$.

We evaluated three standard machine learning algorithms, linear SVM, logistic regression, random forest. We showed that for grab-and-run theft detection, random forest has 0 false positives and 95% true positive rate.

ACSAC'17, San Juan, Puerto Rico, USA

 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

2 RELATED WORK

3 METHODOLOGY

3.1 Data Collection

3.1.1 Experiments. In order to collect positive samples for the machine learning algorithms, we carefully designed experiments to simulate three types of theft scenarios. In these experiments, one researcher acted as a smartphone user, and another one played the role of a thief. In the first experiment, the user stood still and held the phone in front of her as if she was using the phone, for instance texting; the thief approached her from behind, grabbed the phone from her hand and ran away in the forward direction. In the second experiment, the user again was holding the phone in front of her while she was walking at a constant speed; the thief approached her from behind, grabbed the phone and ran away in the forward direction. The third experiment simulated pick-pocket theft. The user put her phone in the back pocket and stood still; the thief approached her from behind, stole the phone from user's pocket and ran away in the forward direction.

We ran 20 trials for each theft scenario, and between two consective trials, researchers put the phone down on the ground for 30-50 seconds which created gaps to help seperate trials. We ran those three experiments at two different times, each of which consisted of 30 trials and lasted approximately one hour. We chose to run the experiments on a flat ground at an open space, so the experiments were not be interrupted. We also made sure that the thief could run at least 40 feet after gaining possession of the victim's phone.

3.1.2 User Study. We conducted a user study in the Bay Area in the United States from September to December 2016 to collect smartphone sensor data, including accelerometer, step count, ambient light, etc, during participants' daily usage. The accelerometer data was then used to generate negative samples for the machine learning algorithms.

We first posted a recruitment advertisement on the Craigslist under the SF Bay Area 'et cetera jobs' category in September 2016 (See Appendix A for recruitment advertisement). All subjects were required to take an online screening survey, in which they provided information about their age, gender, smartphone maker and model, the way to carry a phone, e.g. in a pocket, and whether they are comfortable with wearing a smartwatch. We only recruited participants who used an Android phone with version 5.0 and above, and were comfortable with wearing a smartwatch.

All qualified participants were scheduled via email a 30-minute meeting with a researcher. During the visit, they were instructed to install an application, which collected and transmitted sensor data from their phones to a private account on a cloud server, on their smartphones. Researchers also provided each participant with a Basis Peak smartwatch and explained that they might experience overheat while wearing it. And if it did, they should take it off and contact us immediately. In addition, they signed a consent

form, which explained the purpose, requirements, risks, confidentiality and compensation of the study. Participants then received $25 Visa gift card.

ToDo: participants' demographic information

We had a total of 3 rounds; each round lasted 3 consecutive weeks. A total of 55 participants were recruited. 53 out of the 55 subjects completed the study. In the first round, 16 out of 18 participants finished the study. All 18 subjects who participated in the second round completed the study. So are all 19 participants in the third round.

During the user study, we asked participants to wear a smartwatch for as long as possible except while sleeping. The smartwatch was connected to the smartphone via bluetooth, which allowed us to know the ground truth of the phone being in possession of the user. All log files that contained sensor data were encrypted using AES/CCM with an 11 byte Nonce and a 16 byte MAC upon hourly upload to a cloud server. Researchers contacted participants weekly to make sure that their phones and watches functioned correctly.

At the end of the study, participants returned the watch, filled out a short exit survey and received compensation of $125 Visa gift card.

3.2 Training Set Generation

In order to select features used to generate positive and negative training samples, we first came up with candidate features, minimum, maximum, mean, standard deviation, root mean square, arc length, arc length times standard deviation, and mean absolute of the x, y, z, and magnitude of accelerometer data in one-second windows. After observing plots of positive and negative raw accelerometer data versus time, we decided to use magnitude exceeding 40 m/s^2 as a trigger condition. In other word, we only calculate features vector of one-second window before and after each spike in magnitude of the accelerometer data. Finally, we selected maximum, mean, standard deviation, root mean square, arc length, arc length times standard deviation, and mean absolute. And those features were computed for the magnitude of accelerometer data because compared to x, y, z accelerometer data, the magnitude is more robust to the direction change and performed better. We removed minimum from the feature sets because it did not seem to provide useful infomation for the classifiers. As a result, we generated 60 positive data points and 248508 negative data points. Each data point is a 12 dimentsional feature vector.

3.3 Machine Learning Algorithms

We tried three standard machine learning algorithms, random forest, logistic regression and linear SVM, provide by the Python scikit-learn library, to classify binary classes, theft and normal usage. In order to mitigate the class imbalance problem, i.e. we have much more negative samples than the positive ones, we used the class weight class attribute built in the library and experimented different ratio of positive and negative class weights.

4 RESULTS

We ran a 10-fold cross validation on the entire data set, which consisted of 60 positive samples and 248508 negative samples. Among the three machine learning algorithms we chose, random forest with ratio of positive and negative class weights as 1 performs the best in term of low false negative and false positive rate. Confuction matrices of logistic regression with balanced class weight, i.e. ratio of positive class weight to negative class weight is 4142, and linear SVM with ratio of positive and negative class weights as 1 are also shown as comparisons,

		Predicted Labels	
		Negative	Positive
True Labels	Negative	248508	0
	Positive	3	57

Confusion Matrix of Random Forest

		Predicted Labels	
		Negative	Positive
True Labels	Negative	247370	1138
	Positive	2	58

Confusion Matrix of Logistic Regression

		Predicted Labels	
		Negative	Positive
True Labels	Negative	248497	11
	Positive	30	30

Confusion Matrix of Linear SVM

The random forest classifier has a false negative rate of 0, which means that 0 false alarms would occurr during 3 weeks of user study, and a true positive rate of 95%. In addition, the logistic regression classifier has a false negative rate of 0.46%, which means that on average each user would receive 6.897 false alarms every week, and a true positive rate of 96.7%.

The most predictive features are the maximum and standard deviation of the magnitude of the accelerometer data from the one-second window before the 40-spikes.

ToDo: why feature ranking

5 DISCUSSION