WebLogger: Stealing Your Personal PINs Via Mobile Web Application

Rui SONG*, Yubo SONG*, Qihong DONG*, Aiqun HU* and Shang GAO†
*School of Information Science and Engineering, Southeast University
†Department of Computing, The Hong Kong Polytechnic University
ruisong20@gmail.com, {songyubo, dongqihong, aqhu}@seu.edu.cn, cssgao@comp.polyu.edu.hk

***Abstract*—In recent years, various sensors have been integrated into smartphones to sense the slight motions of human body. However, security researchers found that these sensors can not only be used in motion detection, but also as side-channel to reveal users' privacy data by inferring keystrokes. What is worse, as defined in W3C specifications, the mobile web applications can get these sensor readings silently without permissions from users. Therefore, when cross-site scripting vulnerabilities are found in a mobile web application, attackers can get users' privacy data remotely via these sensors in theory. However, these attacks are difficult to achieve by the fact that mobile web applications can only get sensor readings with low sampling rate in practical uses. In this paper, we proposed a novel ensemble learning algorithm based on weighted voting to improve the keystroke inferring accuracy in low sensors sampling rate. Based on this novel learning algorithm, a prototype system named WebLogger is developed to demonstrate the possibility of inferring the PIN numbers or passwords entered by mobile phone users from mobile web application silently. The results of experiments show that the prediction accuracy of our learning model can be improved to 70%, which is better than 50% in single machine learning algorithms.**

I. Introduction

Motion sensors embedded in smartphones innovate the interaction with the devices and bring more features to consumers. Sensors like gyroscope, accelerometer and orientation sensor are designed to monitor users' location, movement, orientation, attitude and other information. However, researches have confirmed that motion sensors can act as a side-channel for inferring the users' keystroke or input information on smartphones[1].

Researches have proved that there is correlation between the keystroke events and the motion sensor data patterns[2]. Therefore, some applications are designed by attackers to collect data from motion sensors on different platforms like iOS and Android, and perform inference attacks with machine learning algorithms. These systems use motion sensor readings to infer number sequences[3] or long texts like messages and emails[4], [5]. However, this kind of attacks is limited on a specific operating system and will only be executed when victims actively download and run them.

Attack system on web platform can compensate for this defect to some extent, since it can be executed without downloading at almost every operating system. Combining with cross-site scripting (XSS) attacks, malicious code can be embedded into the websites browsed by victims, and steal their data unconsciously.

The World Wide Web Consortium (W3C) provides a series of APIs for web applications to get access to mobile sensors[6]. W3C specifications consider the security and privacy issues on a variety of mobile sensors, and divide these sensors into several security levels[7]. W3C thinks that motion sensors and orientation sensors are less sensitive than other sensors like geolocation sensors because of the low sampling rate and the low amount of information. Therefore, it fails to discuss the risks and the vulnerabilities we mentioned above. By using the APIs provided by W3C, JavaScript code can get access to motion sensor readings from mobile browsers, and this progress do not need any user permissions.

However, a big challenge in this process is that the sampling rate of sensors in web applications are relatively lower than client applications, which greatly decrease the performance of inference systems. Previous researches tend to use single machine learning algorithm such as neural network for data analysis, which cannot handle raw data with low sampling rate. To address this challenge, a weighted voting algorithm can be introduced to improve the performance of inference attack. This algorithm is partly inspired by ensemble learning theory, which combines the advantages of multiple machine learning algorithms by applying weighted voting on the analysis of several base classifiers. By assigning weights based on the prediction accuracy of base classifiers to each algorithm, this algorithm can get greater performance than single learning algorithm or traditional ensemble learning methods.

In this paper, we introduce WebLogger, a system which infers the PINs or passwords entered on websites. Our WebLogger system contains a JavaScript-based web application which perform attack on mobile web browsers. This application can be embedded into normal web pages and steal data from motion sensors when users input PINs or other number sequences on websites. To overcome the weakness of low sampling

978-1-5386-2062-5/17/$31.00 ©2017 IEEE

rate, WebLogger applies multiple machine learning algorithms, and introduces weighted voting algorithm to improve the performance of the system.

The key contributions of this work are summarized as follows:

- We design a JavaScript-based web application to steal sensor data when users input PINs or passwords. This web application uses the standard APIs from W3C to get access to motion sensors, and can be executed on almost every mainstream mobile browser. This application can support large-scale simultaneous access request with the help of Node.js.
- We introduce a weighted voting algorithm to improve the performance of inference system. By combining multiple algorithms with a weighted voting mechanism, this method can improve the prediction accuracy of the classifier significantly.
- We simulate the scene of password input in people's daily life, and collect about 276,000 labelled sensor data from Android and iOS devices corresponding to 3,000 tap events. In the test of WebLogger, we get 61.5% inference accuracy on Android device and 70.2% on iOS device.

The rest of the paper is organized as follows. In section 2, we discuss previous related work in this field. In section 3, we introduce the weighted voting algorithm used in WebLogger. In section 4, we present the design of our WebLogger system in detail. In section 5, we introduce the results of the tests of WebLogger, and evaluate the performance of this system. In section 6, we discuss some methods to improve our work and some possible countermeasures against the attack. In section 7, we conclude this paper finally.

II. Related Work

Attacks based on side-channels have been researched for a long time on PC and all kinds of smart devices. On traditional PC platform, inference systems have been designed with side-channels like electromagnetic leakage[8] or acoustic signals[9]. On wearable equipment like smart watches, researchers tend to take advantages of sensors embedded to detect the movement of users and infer the input outside the devices. Wang et al. use accelerometer, gyroscope and magnetometer to derive the moving distance of users hand consecutive key entries[10]. Their research reveals the feasibility to infer the passwords when users input on ATM keypads or regular keyboards.

Attacks on smartphones based on mobile sensors have been focused for a long time. Early researchers perform their attacks with camera[11], geolocation sensors[12] or microphone[13]. However, these researches can only exist at the theoretical level now, since access to sensitive sensors are strictly limited by the privilege management mechanism, and third-party applications cannot reach these hardware resources without users' permission.

Therefore, researchers turn to look for other side-channels to build their inference system. Motion sensors begin to be widely used in this field since access to these sensors is not limited by the security mechanisms. Cai et al. prove the feasibility of constructing inference systems based on motion sensors[1]. They use data from orientation sensor to infer the keystrokes on a number-only keyboard and get 70% inference accuracy in landscape mode. Xu et al. use similar principles to infer the users lock screen password. They use the data from accelerometer to detect keystroke events and use the data from orientation sensor to infer the password typed by users[3]. Specifically, their data analysis progress is executed on mobile phones. In addition to numbers, some researcher focuses on characters or long text input. Owusu et al. use accelerometer to infer characters, and confirm that accelerometer data can reduce the search space of character keystroke[2]. Ping et al. try to infer long text like mail messages or tweets[5]. They use machine learning algorithms to roughly predict the input text and then use linguistic models to further correct the wrong predictions. They gain accuracy of about 30% after machine learning and improve the accuracy to 65% after using the linguistic models.

However, client applications on smartphones have inherent drawbacks, because these applications cannot be executed unless users download them to their own devices, and these applications can only run on specific operating system. Thus, some researchers tend to design web applications to circumvent these problems. Mehrnezhad et al. propose PINlogger.js to steal users' privacy when users input PINs on web browsers[14]. However, their one-phase prediction accuracy is not ideal on some devices because of the sampling rate restriction on motion sensors. In this paper, WebLogger is designed to overcome this problem and improve the prediction accuracy on web platforms by introducing the weighted voting algorithm in model training phase.

III. Weighted Voting Algorithm

In WebLogger, a weighted voting algorithm is introduced into training process to improve the prediction accuracy on web platforms. A series of classification models can be generated by putting feature sets into different learning algorithms. Weighted voting can then be executed on these models and get a final classifier with better performance than former ones.

In this algorithm, a series of classifiers are trained by several algorithms firstly. The algorithms selected should have diversity on their basic principles, since heterogeneous algorithms can improve the performance of the final model[15]. These classifiers are then combined by a voting mechanism, where feature sets

are put into classifiers, and the final classification results is decided by weighted voting of base classifiers.

The key to weighted voting algorithm is the setting of weights during voting. In traditional ensemble learning, the combination rules are mainly simple majority, maximum or median of the probabilities. Weighted voting algorithm sets the weights according to the basic prediction accuracy. Specifically, the setting of weights is based on the following criteria:

- The weight b_i is proportional to $log\frac{p_i}{1-p_i}$, while p_i is the prediction accuracy of the corresponding classifier.
- The sum of the weights b_i should be 1.

The reason for setting the weights b_i in this way will be illustrated next. For featured data x, $S = \{s_1, s_2, \ldots, s_n\}$ represents classes which are classified by candidate algorithms $L = \{l_1, l_2, \ldots, l_n\}$, while s_i belonging to class set $C = \{c_1, c_2, \ldots, c_t\}$. The optimal decision function of class c_j in C can be expressed as:

$$g_j(x) = log[p(c_j)p(s|c_j)], j = 1, 2, \ldots, t \quad (1)$$

Due to the independence of conditional probability:

$$\begin{aligned} g_j(x) &= log[p(c_j)\prod_{i=1}^{n} p(s_i|c_j)] \\ &= log[p(c_j)] + log[\prod_{i,s_i=c_j} p(s_i|c_j) \prod_{i,s_i\neq c_j} p(s_i|c_j)] \\ &= log[p(c_j)] + log[\prod_{i,s_i=c_j} p_i \prod_{i,s_i\neq c_j} (1-p_i)] \\ &= log[p(c_j)] + log[\prod_{i,s_i=c_j} \frac{p_i}{1-p_i} \prod_{i=1}^{n}(1-p_i)] \\ &= log[p(c_j)] + \\ &\quad \sum_{i,s_i=c_j} log\frac{p_i}{1-p_i} + \sum_{i=1}^{n} log(1-p_i) \end{aligned} \quad (2)$$

If the prediction result of each classifier is defined by:

$$d_{ij} = \begin{cases} 1 & x\ is\ classifiered\ as\ c_j\ by\ l_i \\ 0 & otherwise \end{cases} \quad (3)$$

Then the decision function $g_j(x)$ can be expressed by:

$$g_j(x) = \sum_{i=1}^{n} b_i d_{ij} \quad (4)$$

Since the item $\sum_{i=1}^{n} log(1-p_i)$ in equation 2 has nothing to do with the decision of the classifier, the decision function can be rewritten as:

$$g_j(x) = log[p(c_j)] + \sum_{i=1}^{n} d_{ij} log\frac{p_i}{1-p_i} \quad (5)$$

Comparing with the decision function in equation 5, b_i in equation 4 should satisfies the following constraint:

$$b_i \propto log\frac{p_i}{1-p_i} \quad (6)$$

Weighted voting can be proved to improve the prediction accuracy through rigorous mathematical deduction. Take $n = 3$ as an example. We can assure that $b_1 \leq b_2 \leq b_3$, and it is easy to prove that there must be one valid expression in $b_1 \leq b_2 \leq \frac{1}{3} \leq b_3$ and $b_1 \leq \frac{1}{3} \leq b_2 \leq b_3$. The situation when first expression is valid will be focused next, since the second situation can be proved in the same way.

If $b_1 + b_2 \leq b_3$, the prediction accuracy should be:

$$\begin{aligned} &p_1p_2p_3 + (1-p_1)p_2p_3 + \\ &p_1(1-p_2)p_3 + (1-p_1)(1-p_2)p_3 = p_3 \end{aligned} \quad (7)$$

The prediction accuracy in this situation is equal to the accuracy of best single classifier.

If $b_1 + b_2 > b_3$, we can get:

$$log\frac{p_1}{1-p_1} + log\frac{p_2}{1-p_2} > log\frac{p_3}{1-p_3} \quad (8)$$

based on expression 6. Expression 8 can be written as:

$$\frac{p_1}{1-p_1} \cdot \frac{p_2}{1-p_2} > \frac{p_3}{1-p_3} \quad (9)$$

Comparing the accuracy of weighted voting with best single base classifier:

$$\begin{aligned} diff &= [p_1p_2p_3 + p_1p_2(1-p_3) + p_1(1-p_2)p_3 \\ &\quad + (1-p_1)p_2p_3] - p_3 \\ &= \prod_{i=1}^{3}(1-p_i)(\frac{p_1}{1-p_1} \cdot \frac{p_2}{1-p_2} - \frac{p_3}{1-p_3}) \\ &> 0 \end{aligned} \quad (10)$$

Therefore, the prediction accuracy after weighted voting is proved higher than single base classifier.

IV. Implementation

The design and implementation of WebLogger will be presented in this section, and the challenges to be addressed in this system will also be revealed. The implementation of WebLogger can be divided into several parts, including data collection, pre-processing, feature extraction and machine learning.

A. Data Collection

To collect motion sensor data from mobile devices, a web application which is based on JavaScript is designed. This web application can be divided into two parts: the client application and the backend server. The client application shows random number sequences of 0 to 9 to the users and activates soft keypad for entering the corresponding number. During the keystroke events, the application collects data from accelerometer, gyroscope and orientation sensor and send the data to the server. The server program gathers the data from clients and save the data into the database. The backend server is designed by Node.js since it can support large-scale simultaneous access request and handle I/O intensive operations.

TABLE I: Features extracted from the raw data.

Feature	Description
Max	Maximum value
Min	Minimum value
RMS	Root-mean-square value
RMSE	Root-mean-square error
Mean	Average value of each tap
PeakNum	Number of local peaks
CrestNum	Number of local crests
SMA	Signal magnitude area
Skewness	Asymmetry of the curve
Kurtosis	Peakedness of the curve
ATP	Average time to a peak
ATC	Average time to a crest

B. Feature Extraction

To construct the training sets for the machine learning, Features should be extracted from the raw data. The raw data which come from 3 motion sensors can be divided into 4 attributes: acceleration, acceleration with gravity, rotation and orientation. Each attribute has 3 data sequences from each axis. The Euclidean norm of each attribute is also calculated since it can represent the total energy of force applied to the devices. Statistical variables in Table I are selected for feature extraction based on some previous work[2]. Each feature in Table I is extracted from every sequence of attributes, which contributes to a 192-dimensional vector for each keystroke event.

C. Model Training

Several machine learning algorithms is introduced to train the base classifiers from the feature vectors:

- SMO, which is the sequential minimal optimization algorithm, is an algorithm based on support vector machine(SVM). This algorithm deconstructs the whole linear programming task into a few of simple classification problems, and then solve these problems by divide and conquer.
- K-NN, which is the k-nearest neighbor algorithm, is a type of instance-based learning. It does not generate model at the training process. Classification will be made during the validation process.
- C4.5, which is a decision tree model based on ID3 algorithm. The design of this algorithm is inspired by the information theory, and its splitting criterion of the branches is the normalized information gain.
- LMT, which is the logistic model tree, is also an algorithm based on decision tree. It classifies the leaf nodes by logistic regression function.
- Random Forest, which is a learning method based on decision tree. This algorithm constructs a series of decision trees and output the predictions by a majority vote of the decision trees.

Weighted voting is performed after training models by the algorithms above to improve the performance of the inference system. It has been proved in section 3 that the weighted voting algorithm can improve the prediction accuracy of the classifier. Three algorithms which has better performance than others are chosen as base classifiers for weighted voting.

TABLE II: Information of the devices in experiments.

Device	iPhone 6s	MI 3
Operating System	iOS Ver. 10.3.1	Android Ver. 4.4.4
Browser	Mobile Safari Ver. 10.0	Chrome Ver. 53.0.2785.146
Browser Engine	WebKit Ver. 603.1.30	WebKit Ver. 537.36

V. Evaluation

In this section, we evaluate the inference performance of our system from several aspects. Experiments was performed on several devices which belong to different operating systems. The details of the devices in the experiments are listed in Table II.

Two volunteers are invited to participate in experiments, both of whom are university students. They are instructed to click on the key on the soft keyboards corresponding to the random number sequences displayed on the screen. To eliminate the interference of other factors, volunteers are required to maintain a normal body posture and avoid sharply body swing during input. During the experiment, 300 input events are recorded corresponding to 3,000 keystrokes. 100 of them come from Android device while the rest 200 come from the iOS device. Nearly 280,000 labelled motion sensor readings are collected in total during the whole process.

A. Accuracy Rate before introducing weighted voting

Classifiers are trained with the algorithms mentioned in section 4. During the training and validation phase, 10-fold cross-validation is performed to evaluate the prediction accuracy of each algorithm precisely. The accuracy rates of each algorithm are illustrated in Figure 1. The performance of LMT is the best of the candidate, while random forest and SMO also get results close to LMT. In contrast, the other two algorithms perform poorly.

However, the time consumption of LMT is much greater than other algorithms during model training process. Figure 2 shows the time consumption of each algorithm when model training and validation. Under the same hardware condition, LMT takes much longer than other algorithms, while Random Forest and SMO can reach a compromise between the classification accuracy and the time cost. These two algorithms can exchange small accuracy loss for a large time advantage.

The sampling rate of raw data collected from motion sensors by web application is relatively lower

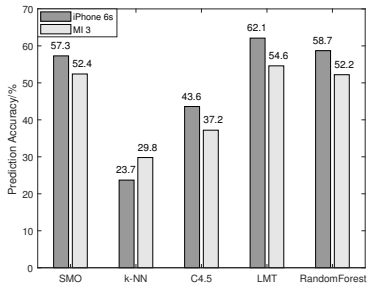

Fig. 1: Accuracy of several machine learning algorithms

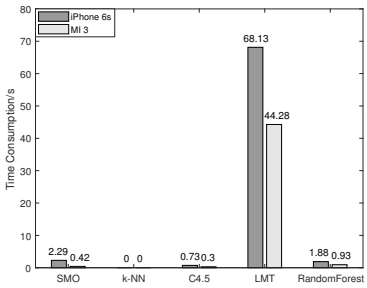

Fig. 2: Time consumption of several machine learning algorithms

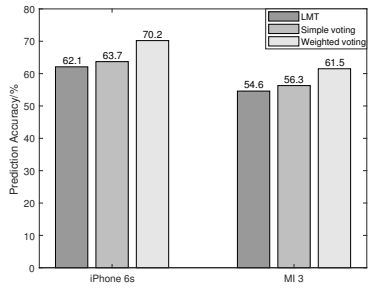

Fig. 3: Accuracy rate with different methods

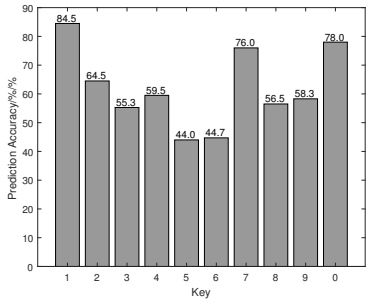

Fig. 4: Accuracy of keys in LMT algorithm

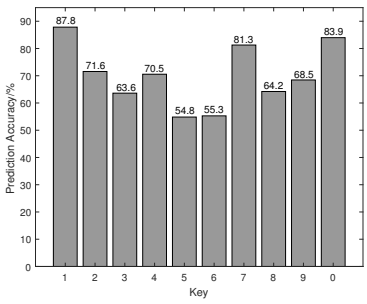

Fig. 5: Accuracy of keys in weighted voting

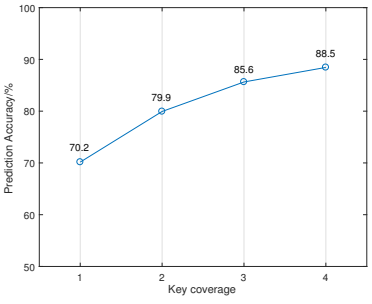

Fig. 6: Prediction Accuracy of different digits coverage.

than those from native applications. Specifically, in the experiment on WebLogger, the sampling rates of iPhone 6s and Xiaomi Mi 3 are about 120Hz and 85Hz respectively. The low sampling rate deteriorates the prediction accuracy of WebLogger, which calls for a method to solve this dilemma.

B. Accuracy Rate after introducing weighted voting

Weighted voting is introduced in WebLogger to compensate for the effect from low sampling rate. Figure 3 shows the inference accuracy when introducing weighted voting for model training. Figure 3 reveals that the prediction accuracy of both devices is improved by the weighted voting algorithm.

Both simple voting method and weighted voting method are performed for model training respectively. Figure 3 indicates that simple voting method inspired by traditional ensemble learning theory has little effect on the performance of classifier, while the weighted voting method can achieve better inference accuracy than simple voting. By combining the multi-dimensional features of different classifiers, weighted voting can complement the defects of individual classifiers and improve the overall classification accuracy.

When considering the prediction accuracy by keys, weighted voting has different effect on the performance of each key, and the difference shows a certain distribution. Figure 4 shows the prediction accuracy of each key when using LMT, while the prediction accuracy of keys on the edge of the keypad are obviously higher than other keys, and the accuracy of keys on the right side is generally lower than the keys on the left side. It is probable that the volunteers habits of using smartphones leads to such a result. Figure 5 shows the prediction accuracy when using weighted voting method, while the keys with poor prediction performance in former algorithms get significant improvement. This result proves that the weighted voting algorithm can effectively improve the prediction accuracy of poor performance classes in weak base classifiers.

C. Prediction Sequence

Since the keys on keypads are very close to each other, each key has a great chance of being mistaken as its adjacent keys. However, such a false inference is not entirely worthless. A prediction sequence for each key can be constructed based on the confusion matrix of the prediction result. Table III lists several potential possible predictions for each key.

When inferring the input of a numerical sequence, this table can be used to reduce the searching space. Figure 6 shows the prediction accuracy when cover different digits in an inference system. Most keys can achieve an extremely high accuracy when increasing the prediction digits to 4. With the help of the prediction sequence, the searching space will be reduced effectively, while the average search times of a 4-digit numerical password can be reduced to 2.6% of the original.

TABLE III: Ranked prediction table, we list 3 most confused numbers for each key.

Key	True Value	2nd	3rd	4th
1	87.8	**4**: 4.5	**2**: 1.2	**5**: 0.5
2	71.6	**5**: 9.7	**4**: 6.5	**3**: 3.4
3	63.6	**6**: 10.1	**5**: 6.2	**2**: 2.8
4	70.5	**7**: 10.6	**1**: 4.3	**2**: 1.0
5	54.8	**2**: 13.7	**8**: 12.3	**4**: 4.5
6	55.3	**9**: 12.4	**3**: 9.6	**8**: 7.7
7	81.3	**4**: 6.1	**0**: 2.8	**2**: 0.5
8	64.2	**5**: 11.7	**9**: 6.4	**6**: 3.7
9	68.5	**6**: 12.1	**3**: 6.0	**8**: 3.4
0	83.9	**8**: 6.9	**7**: 1.7	**9**: 0.6

VI. Discussion

A. Future Work

The results of experiments show that WebLogger can infer the keystroke on web browsers effectively. However, our system still has room to be improved. Our WebLogger only focuses on the inference of numerical sequences. However, there is also a lot of text input on smartphones in peoples daily lives. Actually, most part of WebLogger can be used for text inference. The prediction sequence is also very suitable for long text inference since there are inherent logical associations between the characters in words. By introducing the techniques of natural language processing, WebLogger can be improved for text input prediction.

B. Countermeasures

The experiments on WebLogger show the feasibility of stealing users' input information. With the rapid adoption of smart devices, the vulnerabilities revealed in our research will be exploited by attackers for illegal activities. Next, some countermeasures to mitigate this threat will be discussed.

The success of keystroke attacks shows that the privilege management of mobile operating systems needs to be improved. Our experience shows that the sampling rate of the motion sensors can influence the prediction accuracy of inference systems. Thus, it is useful to reduce the sampling rate on background applications and web browsers to reduce the impact of this kind of attacks. In addition, injecting noise signal to motion sensors can also be a help. Some researchers have considered this method to ease the attack[16]. They have proved that this method can reduce the risk of keystroke leakage and will not affect the benign applications.

VII. Conclusion

In this paper, we introduce WebLogger for PINs and passwords inference on web browsers of smartphones based on motion sensor readings. Specifically, we design a data collection application based on JavaScript code to collect the data from accelerometer, gyroscope and orientation sensor, and then extract feature sets from the raw data. A weighted voting algorithm is introduced in the process of classification model training, and improves the inference accuracy significantly. The results of experiments show that WebLogger can effectively infer the PINs and passwords input on browsers of smartphones.

VIII. Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities(NO.2242017K40013).

References

[1] L. Cai and H. Chen, "Touchlogger: inferring keystrokes on touch screen from smartphone motion," in *Usenix Conference on Hot Topics in Security*, 2011, pp. 9–9.

[2] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 9.

[3] Z. Xu, K. Bai, and S. Zhu, "TapLogger: inferring user inputs on smartphone touchscreens using on-board motion sensors," in *ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2012, pp. 113–124.

[4] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *International Conference on Mobile Systems, Applications, and Services*, 2012, pp. 323–336.

[5] D. Ping, X. Sun, and B. Mao, "TextLogger: inferring longer inputs on touch screen using motion sensors." ACM Press, 2015, pp. 1–12.

[6] F. Hirsch, "Device and sensors working group," World Wide Web Consortium, Tech. Rep., 2017. [Online]. Available: https://www.w3.org/2009/dap/.

[7] K. R. Christiansen and A. Shalamov, "Motion sensors explainer," World Wide Web Consortium, Tech. Rep., 2017. [Online]. Available: https://www.w3.org/TR/2017/NOTE-motion-sensors-20170511/.

[8] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards." in *Proceedings of the 18th conference*. USENIX security symposium, 2009, pp. 1–16.

[9] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 245–254.

[10] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN." ACM Press, 2016, pp. 189–200.

[11] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, "Stealthy video capturer: a new video-based spyware in 3g smartphones," in *Proceedings of the second ACM conference on Wireless network security*. ACM, 2009, pp. 69–78.

[12] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, 2009, pp. 31–36.

[13] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, and A. Kapadia, "Soundminer: A stealthy and context-aware sound trojanfor smartph-ones."

[14] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "Touchsignatures: identification of user touch actions and pins based on mobile sensor data via javascript," *Journal of Information Security and Applications*, vol. 26, pp. 23–38, 2016.

[15] T. Dietterich, "Ensemble methods in machine learning," *Multiple Classifier Systems*, pp. 1–15, 2000.

[16] P. Shrestha, M. Mohamed, and N. Saxena, "Slogger: Smashing Motion-based Touchstroke Logging with Transparent System Noise," in *ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 67–77.