Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2016, Article ID 6860217, 6 pages
http://dx.doi.org/10.1155/2016/6860217

Research Article

Power Consumption Based Android Malware Detection

Hongyu Yang and Ruiwen Tang

School of Computer Science and Technology, Civil Aviation University of China, No. 2898, Jinbei Road, Tianjin 300300, China

Correspondence should be addressed to Hongyu Yang; yhyxlx@hotmail.com

Received 11 November 2015; Revised 9 March 2016; Accepted 27 March 2016

Academic Editor: Anthony T. S. Ho

In order to solve the problem that Android platform's sand-box mechanism prevents security protection software from accessing effective information to detect malware, this paper proposes a malicious software detection method based on power consumption. Firstly, the mobile battery consumption status information was obtained, and the Gaussian mixture model (GMM) was built by using Mel frequency cepstral coefficients (MFCC). Then, the GMM was used to analyze power consumption; malicious software can be classified and detected through classification processing. Experiment results demonstrate that the function of an application and its power consumption have a close relationship, and our method can detect some typical malicious application software accurately.

1. Introduction

Smart phones and other mobile terminals have become the most important part of modern life, and Android operating system in the smart phone has more than 50% share. However, not only do open source characteristics of the Android platform provide convenient conditions for software developers, but also security problem has become a hot spot of concern. According to the latest statistics of F-Secure [1], in 2014 the Android platform has 277 species threats, such as unauthorized download applications and GPS location tracking. So distinguishing the mobile normal software and malicious software is a hot problem to be solved.

At present, there are two kinds of methods for malware detection: one is static analysis method, and the other is dynamic analysis method. The static analysis is mainly based on application decompiled and gets source code of malicious behavior.

However, due to the limitation of software code analysis method and decompiled technology, dynamic analysis has gradually become the trend of the mobile malware detection technology development.

In dynamic analysis field, Zheng et al. [2] propose a dynamic analysis system running in the Dalvik virtual machine, but because of the limitation of Dalvik layer in Android, it cannot monitor the detailed information of underlying code. Enck et al. [3] set up a monitoring program running in simulation environment; this method can effectively achieve real-time behavior for a specified application, but it cannot run on Android devices without ROOT access.

For the Android platform application software, there are two ways to get power consumption information. We can use native Android API to calculate power consumption. This method has been used by some application software, such as PowerTutor [4].

Another method is proposed by Curti et al. [5] who rewrite the underlying driving mode for power information. Compared with the first method, this method is more accurate, but it has high intervention to the system.

In the study of power consumption analysis method to detect malicious software, Jacoby et al. [6] propose a model which is based on power consumption to detect intrusion.

Based on this, Buennemeyer et al. [7] and Kim et al. [8] use the Android mobile phone battery configuration files to detect malicious software. But all of the above methods only consider the amount of power consumption, without taking the timing characteristics of power consumption into account.

According to the above problem, this paper puts forward a kind of mobile malware detection model based on mobile phone power consumption. The model structure is shown as in Figure 1, and it mainly includes four function parts:

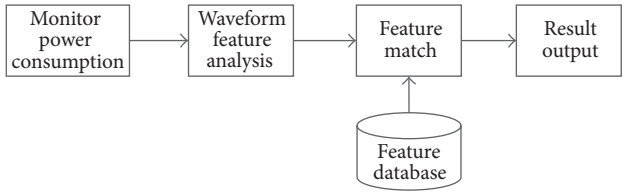


Figure 1: Malware detection model.

(1) Battery consumption monitor: monitoring of the application software in a certain time range of power consumption.

(2) Analysis waveform characteristics: analysis of the diagram of power consumption and extraction of the feature of power consumption waveform.

(3) Waveform feature matching: matching the feature of power consumption waveform which is calculated by the analysis module in the prebuild feature database.

(4) Output results: output of the classification results for each power consumption feature.

2. Battery Power Consumption Monitoring

The method for monitor battery power consumption is the key to this detection model; if battery power consumption data is not accurate, it will directly affect the accuracy of the results. In this paper, we use PowerTutor [4] to monitor battery power consumption information of the specified application.

PowerTutor can get the battery power consumption time-series data of each mobile application by monitoring their components including CPU, audio, screen, Wi-Fi, 3G, and GPS.

During the power consumption monitoring, we found that power consumption has the same state transition characteristics as sound signal did. But the state of power consumption is much more stable, and 5-minute sampling time can cover all status changes. So, in the rest of this paper, for each application we take 5 minutes to monitor battery consumption.

Figure 2 shows the CPU power consumption of two Android applications ("UC browser" and "iReader") within 5 minutes. It can be found from Figure 2 that the consumption curves of two applications with different functions are significantly different. It also shows that we can effectively distinguish different applications through the analysis of the power consumption.

3. Waveform Feature Extraction and Analysis

We know that applications with different functions have different power consumption from Figure 2. But when using these consumption data to classify software, we also need to consider the influence of different factors on the power consumption, such as uncertainty user input, different screen resolution, and the influence of different hardware acceleration technology on power consumption. In order to reduce the above factors on the result of application software to identify or test, it is important to choose an effective method to extract the feature information of power consumption waveform.

In the field of waveform characteristics analysis, the most widely used and the most mature way is Mel frequency cepstrum coefficient (MFCC). It has been widely used in speech recognition and has high identification accuracy [9, 10].

This paper designs a MFCC calculation process for waveform feature extraction which was mentioned before (see Figure 3).

MFCC computation process is as follows.

(1) Frame Pretreatment. Frame pretreatment in this paper only needs to do framing and window processing for the original input data of $S(n)$. The purpose is to extract short signal features (similar to audio processing). In order to facilitate the calculation of subsequent steps, set the frame length 5 s; frame offset (the overlap between two frames) is 2.5 s. During the processing, we use Hamming window; the calculation formula is

$$X(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad (0 \le n \le N-1). \tag{1}$$

Among them, the 0.54 and 0.46 are the constant which the Hamming window defined, N is sampling points per frame, and $X(n)$ is the short signal feature which is the result of the Hamming window formula.

(2) DFT (Discrete Fourier Transform) Linear Spectrum Calculation. We use Fourier transform to calculate the short-time energy and frequency domain signals. First of all, add some 0 after each frame in the time domain signal $x(k)$ to make sure the frame length is N ($N = 256$); then, we can get the linear spectrum $X(k)$ through the discrete Fourier transform. The calculation method is

$$X(k) = \sum_{k=0}^{N-1} x(k)\, e^{-j2\pi_j k/N}, \quad (0 \le k \le N-1). \tag{2}$$

(3) Mel Frequency Filter Calculation. The input frequency signal through a number of triangle pass filters and Mel frequency spectrum signal are obtained by filtering calculation. Among them, each of the triangular filter center frequency calculation steps is as follows.

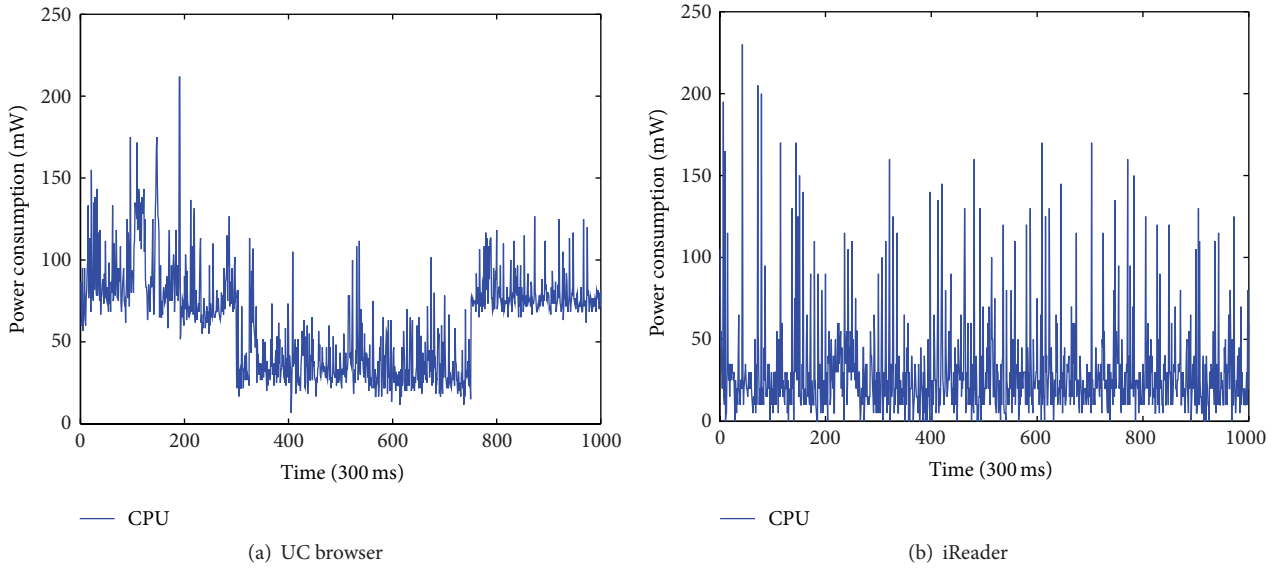
Set the number of filter k, signal sampling frequency is f_s, and the max Mel frequency is

$$f_{\max}[\text{Mel}] = 2595 \times \log_{10}\left(1 + \frac{f_s}{700\,\text{Hz}}\right). \tag{3}$$

Center frequency spacing between each filter is

$$\Delta_{\text{Mel}} = \frac{f_{\max}}{k+1}. \tag{4}$$

(4) Logarithmic Energy Calculation. In order to have better robustness of noise, it is indispensable to calculate the logarithmic energy of Mel frequency spectrum. Set the number



(a) UC browser (b) iReader

Figure 2: Application of battery consumption sequence diagram.

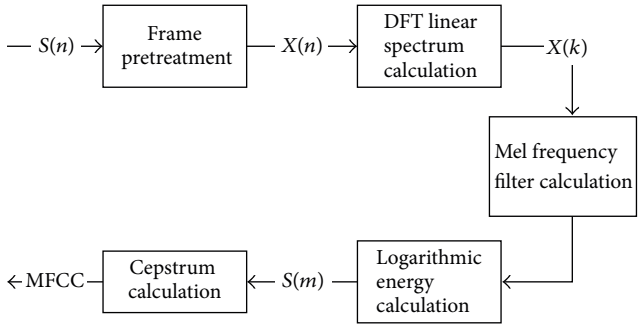


Figure 3: MFCC calculation process.

of triangular filter for M; logarithmic spectrum $S(m)$ of linear spectrum $X(k)$ is calculated as follows:

$$e(m) = \sum_{k=0}^{N-1} |X(k)|^2 H_m(k), \quad (0 \le m \le M)$$
$$S(m) = \ln(e(m)), \quad (0 \le m \le M). \tag{5}$$

(5) Cepstrum Calculation. Through the discrete cosine transform (DCT), we can transfer the logarithmic spectrum $S(m)$ to cepstrum; the formula is

$$C_{\text{Mel}}(n) = \sum_{m=0}^{M-1} S(m) \cos\left(\frac{\pi n (m + 1/2)}{M}\right), \quad (0 \le m < M). \tag{6}$$

4. Waveform Feature Matching

Feature matching includes two processes. Firstly, create the battery power consumption characteristics database. And then match software battery power consumption characteristics of input signal with the feature database. In the feature matching calculation, we choose Gaussian mixture model (GMM) [11]. The advantage of GMM is that it can approximate to arbitrary shape distribution probably and the algorithm complexity is low. Battery power consumption feature matching process design is as follows.

(1) Build Characteristic Database. The key to build battery power consumption characteristics database is essentially to calculate Gaussian distribution parameters of the given MFCC feature sequence. Therefore, we can use the maximum likelihood estimation to obtain probability distribution parameters; calculation method is as follows.

Set the MFCC sequence $X = \{x_t,\ t = 1, 2, \ldots, T\}$. Each component of X is independent of each other; the likelihood for the GMM model parameters is

$$L(\lambda \mid X) = \prod_{t=1}^{T} P(x_t \mid \lambda). \tag{7}$$

The eligible parameters are to find a group $\widetilde{\lambda}$ which makes $L(\lambda \mid X)$ largest:

$$\widetilde{\lambda} = \max_{\lambda} L(\lambda \mid X) = \max_{\lambda} P(X \mid \lambda) \tag{8}$$

or

$$\ell(\widetilde{\lambda}) = \max\left(\sum \log P(x \mid \lambda)\right). \tag{9}$$

While using maximum likelihood estimation, we choose to use the expectation maximization (EM) algorithm [12] to get the estimated parameters. The calculation steps are as follows.

For each training sample i, z is the category which this sample may belong to, and Q_i is a certain probability distribution of z; then

$$Q_i\left(z^i\right) = P\left(z^i \mid x^i; \lambda\right). \tag{10}$$

Estimated parameters are calculated as follows:

$$\theta = \max_{\theta} \sum_i \sum_{z^i} Q_i\left(z^i\right) \log \frac{P\left(x^i, z^i; \lambda\right)}{Q_i\left(z^i\right)}. \tag{11}$$

The EM algorithm can converge because of Jensen's inequality [13].

According to formula (10) and Jensen's inequality, when calculating expectation we assume that the value of λ is unchanged; then we can get the equation

$$\ell\left(\lambda^{(t)}\right) = \sum_i \sum_{z^i} Q_i^{(t)}\left(z^{(i)}\right) \log \frac{P\left(x^{(i)}, z^{(i)}; \lambda^{(t)}\right)}{Q_i^{(t)}\left(z^{(i)}\right)}. \tag{12}$$

But in each iteration, the value of λ is changed; if we treat λ as variable and calculate its derivative, we can get

$$\begin{aligned} \ell\left(\lambda^{(t+1)}\right) &\geq \sum_i \sum_{z^{(i)}} Q_i^t\left(z^{(i)}\right) \log \frac{P\left(x^{(i)}, z^{(i)}; \lambda^{(t+1)}\right)}{Q_i^{(t)}\left(z^{(i)}\right)} \\ &\geq \sum_i \sum_{z^{(i)}} Q_i^t\left(z^{(i)}\right) \log \frac{P\left(x^{(i)}, z^{(i)}; \lambda^{(t)}\right)}{Q_i^{(t)}\left(z^{(i)}\right)} \\ &= \ell\left(\lambda^{(t)}\right). \end{aligned} \tag{13}$$

Through formula (12) and formula (13) we proved λ is monotonically increasing. So one way to convergence is that λ no longer changes, or λ change very little.

(2) Features Value Matching. After building the database, the given X is a known characteristic. For the existing categories z, calculate the probability value $P(x \mid z_i)$. Characteristic value matching is finding z_i, which can make $\max_i P(x \mid z_i)$; then z_i is the output of feature matching.

5. Experiments and Result Analysis

In the experiment, first we use the method proposed in Section 3 to build GMM model for game, browser, music player, and malicious software. And for the malicious software, we simulate a floating point number attack for the Android platform. In the construction of the GMM model, we repeatedly monitor the software's power consumption in 5 minutes and calculate the MFCC features for the reset of 3 types of applications and then use the EM algorithm to calculate the GMM model parameters.

Experiments are designed on 6 cell phones, including Samsung GALAXY S5 and LG G2 mobile phones. Battery monitoring tool is PowerTutor [4]. In the experiments, we only monitor and collect the CPU power consumption data when the application software is running. The cycle time of monitoring is 5 minutes and sampling frequency is 300 ms.

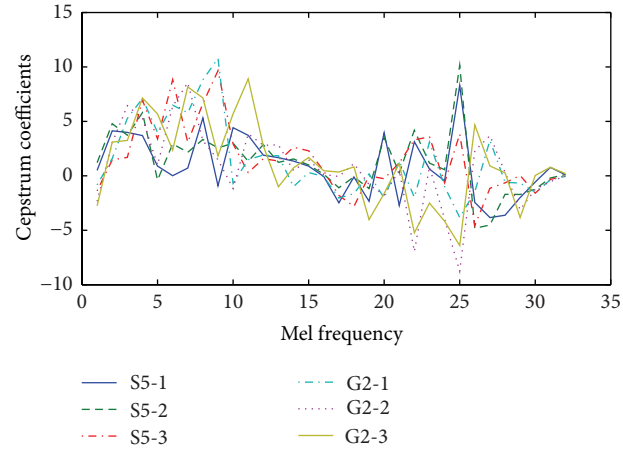


Figure 4: MFCC feature distribution.

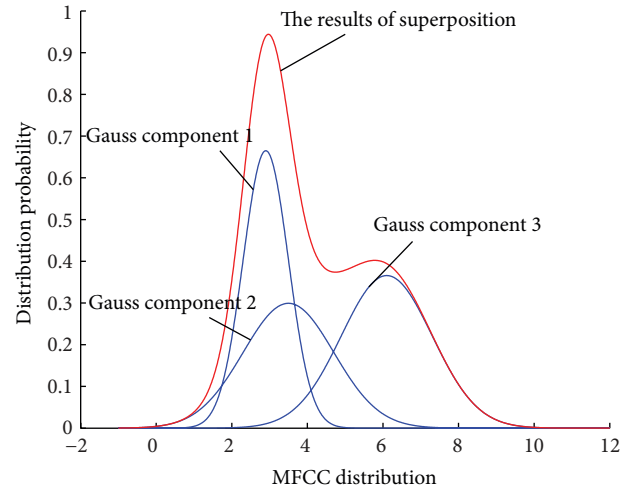


Figure 5: iReader GMM model.

MFCC feature calculation results are shown as in Figure 4; it is the test result of the application that is named iReader. The GMM parameters model is shown in Figure 5.

We can see from Figure 4 although power consumption MFCC feature values for the same application in different mobile phones are not completely the same, the distribution is roughly the same. It also proves that the MFCC feature extracting can largely reduce the equipment and other impacts caused by factors.

And, in Figure 5, the GMM model also shows that the estimated parameters can be calculated using the extracted MFCC parameters and the result is convergence.

In application software testing, firstly we build GMM model for four kinds of typical applications, including games, browser, music player, and malicious software. In this test, this paper only simulates one kind of malicious software which is floating point attack.

During the test step, this paper selects 100 applications in each group. Under the normal condition, we monitor the CPU battery consumption during 5 minutes and then classify it through the proposed power analysis model. Four kinds

Table 1: Typical applications results.

Category	Results			
	Game	Browser	Music	Malicious software
Game	78	9	3	10
Browser	12	72	7	9
Music	16	11	65	8
Malware	3	7	11	79

Table 2: Detection rate of statistical.

Category	Results		
	NR	PR	AR
Game	10%	90%	78%
Browser	9%	91%	72%
Music	8%	92%	65%
Malware	79%	21%	79%

Table 3: Different model orders and results.

Order of GMM	2	3	4	8	12
AR (%)	55.1	65.2	79.7	77.9	66.7

of typical application test results are shown in Table 1, and the detection rate of the four kinds of typical applications is shown in Table 2.

The malware detection rate in Table 2 (negative rate, NR) shows that the proportion of application was classified as malware, benign detection rate (positive rate, PR) shows the proportion of application software was classified as benign, and accuracy rate (AR) shows the proportion of application by accurate classification. Table 2 data show that the proposed power analysis model to differentiate benign software and malicious software is highly accurate.

The experiment proves that CPU power consumption characteristics of simulated malicious software have a significant difference from others, and through the analysis of the power consumption we can detect and classify malicious software.

Since our method is based on the Gaussian mixture model, we know that the model order will affect the test results. So during the experiments, we change the model order from 2 to 12 with the same sampling time and the same sample data of music applications and compare the results. And the experimental results are shown in Table 3.

The data in Table 3 can prove that, with the increase of GMM's order number, detection accuracy has a significant rising. But when the GMM's order is greater than 4, detection precision begins to decline. This is because as the GMM's order is larger the model becomes more complex; the matching process and parameters calculating process will take a longer time. Since our experiment takes 5 minutes' sampling time, when GMM's order becomes larger the sample data is too little to calculate accurately the higher order of GMM's parameters.

6. Conclusions

Based on battery power consumption sequence characteristics of the application software which is collected from mobile terminals such as mobile phone, this paper introduces MFCC feature extraction algorithm and GMM model classification algorithm and then proposes an Android malware detection method. What is more, the experiment proves that the method has effective identification and detection of malicious software.

In the future, we will try to make MFCC parameters extraction and classification of GMM model integrate into the mobile terminal equipment and consider the battery consumption of other hardware when application software is running.

Competing Interests

The authors declare that they have no competing interests.

References

[1] F-Secure, *Mobile Threat Report Q1 2014*, 2014.

[2] M. Zheng, M. Sun, and J. C. S. Lui, "DroidTrace: a ptrace based Android dynamic analysis system with forward execution capability," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '14)*, pp. 128–133, IEEE, Nicosia, Cyprus, August 2014.

[3] W. Enck, P. Gilbert, B.-G. Chun et al., "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI '10)*, pp. 18–21, 2010.

[4] L. Zhang, B. Tiwana, R. P. Dick et al., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '10)*, pp. 105–114, IEEE Press, Scottsdale, Ariz, USA, October 2010.

[5] M. Curti, A. Merlo, M. Migliardi, and S. Schiappacasse, "Towards energy-aware intrusion detection systems on mobile devices," in *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS '13)*, pp. 289–296, IEEE, Helsinki, Finland, July 2013.

[6] G. A. Jacoby, R. Marchany, and N. J. Davis IV, "Battery-based intrusion detection: a first line of defense," in *Proceedings fron the 5th Annual IEEE System, Man and Cybernetics Information Assurance Workshop (SMC '04)*, pp. 272–279, IEEE Press, Piscataway, NJ, USA, June 2004.

[7] T. K. Buennemeyer, T. M. Nelson, L. M. Clagett, J. P. Dunning, R. C. Marchany, and J. G. Tront, "Mobile device profiling and intrusion detection using smart batteries," in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, p. 296, IEEE, January 2008.

[8] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 239–252, ACM Press, New York, NY, USA, June 2008.

[9] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2010.

[10] S. G. Kumar, R. K. Prasad, M. Rao et al., "Speaker recognition using GMM," *International Journal of Engineering Science and Technology*, vol. 2, no. 6, pp. 2428–2436, 2010.

[11] V. Christleig, D. Bernecker, F. Honig et al., "Writer identification and verification using GMM supervectors," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV '14)*, pp. 998–1005, IEEE Press, Piscataway, NJ, USA, March 2014.

[12] Z. Ju, Y. Wang, W. Zeng, H. Cai, and H. Liu, "A modified em algorithm for hand gesture segmentation in RGB-D data," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '14)*, pp. 1736–1742, IEEE Press, Beijing, China, July 2014.

[13] https://en.wikipedia.org/wiki/Jensen%27s_inequality.