Mobile Terminal Security Monitoring System Based on Distributed Agent

Huan Ying[1, a*], Yu Luo[2, b], Lifang Han[1, c], Liang Zhou[1, d], Songling Shan[1, e], and Yanhong Chen[1, f]

[1]China Electric Power Research Institute, Beijing 100192, China

[2]State Grid Henan Information & Telecommunication Company, Zhengzhou 450052, China

[a]yinghuan@epri.sgcc.com.cn, [b]luoyu@ha.sgcc.com.cn, [c]hanlifang@epri.sgcc.com.cn, [d]zhouliang@epri.sgcc.com.cn, [e]shansongling@epri.sgcc.com.cn, [f]chenyanhong@epri.sgcc.com.cn

Keywords: Mobile terminal; Security monitoring; Data loss prevention

Abstract. Aiming at the lack of effective security monitoring of mobile terminals, this paper proposes a mobile terminal security monitoring system based on distributed agent, which adopts distributed deployment and centralized management. Also, a multi-level malicious behavior monitoring model is proposed to monitor the behavior of mobile terminal hierarchically. Based on security monitoring and dynamic disposition, the monitoring system prevents the end-user's sensitive data loss to protect the mobile terminal security.

Introduction

With the development of information network, all kinds of mobile terminals such as mobile personal digital assistant (PDA), mobile smart phones are widely used in various industries. However, such as viruses, worms, Trojans and other malicious code attacks information network through vulnerability of mobile terminal, and it leads to unsafe state for the network.

Nowadays, there are many studies on mobile application behavior detection: [1] presents an Android malicious code detection system based on kernel behavior analysis. The experimental results show that the system can effectively detect unknown malicious behavior; [2] founds that 66% of the regular application access to the user's address book information according to statistical analysis of the Android market;[3] adopts traditional digital forensics technology to remote monitoring and auditing Android smart phone; DexDiff [4] is a system for auditing third-party mobile software which compares pre-installed applications and libraries from the handset with the corresponding base system in the Android open source project library to detect inventory pre-installed applications and potential vulnerabilities in the library;[5] proposes the concept of sensitive path, and it improves the efficiency and accuracy of analysis and detection. However, these methods [6,7] only have in-depth study on mobile terminal malicious behavior analysis, and didn't give a comprehensive mobile terminal security monitoring framework and cannot be directly applied to the mobile business scenarios.

Therefore, this paper presents a mobile terminal security monitoring system model based on distributed agent. The system is based on distributed agent to deploy data collection environment and proposes a multi-level malicious behavior monitoring model. Also, the security of the mobile terminal is guaranteed by data preprocessing, malicious behavior dynamic audit and data loss prevention.

Overview

The architecture of mobile terminal security monitoring system based on distributed agent is shown in Fig. 1, which includes data collection based on distributed agent, dynamic audit of malicious behavior and data loss prevention. The data acquisition module preprocesses the collected monitoring data and stores it into a unified mobile terminal security monitoring information database to provide important information for subsequent analysis. The malicious behavior dynamic audit module executes terminal behavior audit analysis, and intercepts malicious behavior based on

several technologies of behavior feature extraction for mobile terminal application, combined with the business security threat policy library. Also, data loss prevention module is interrelated with dynamic audit module: it identifies sensitive data call behavior in the process of dynamic audit process, while sensitive data call blocking is realized during malicious behavior interception. Finally, combined with data encryption, data desensitization and other technologies the module prevents mobile terminal user's data loss.

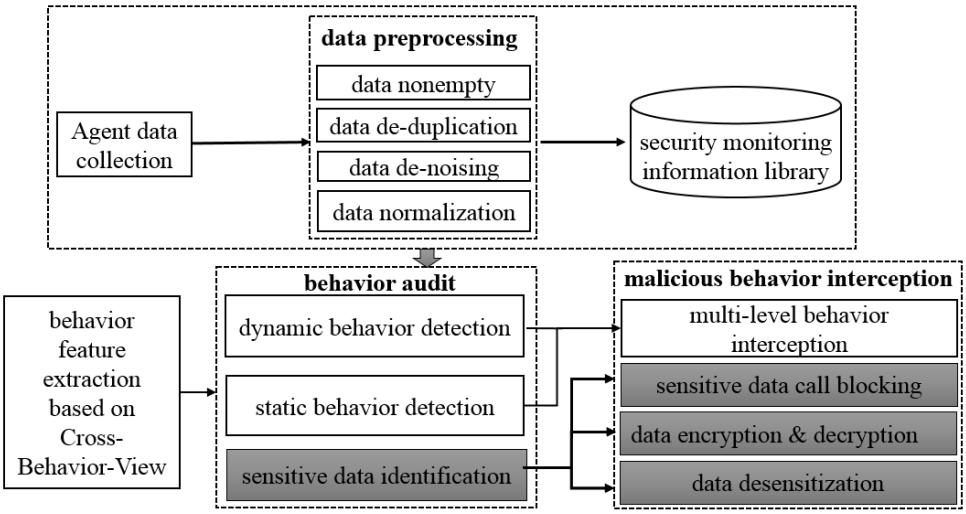

Figure 1. Overview of mobile terminal security monitoring system

Data Preprocessing

The framework of distributed agent data acquisition and preprocessing is constructed to collect the data such as system information and application information stored in mobile terminal, which is suitable for data acquisition requirements of mobile office environment with large-scale mobile terminals. Data preprocessing is essential preparation before data mining. Data extraction, data de-duplication, data de-noising and data normalization are needed to ensure quality and efficiency of data collection.

Malicious Behavior Audit

This paper proposes a multi-level malicious behavior monitoring model from Java level, Native level, Kernel level to achieve effective monitoring and enhance the versatility, hierarchy, fine-granularity of malicious behavior monitoring, so that the hierarchical monitoring of mobile terminal behavior is achieved. The dynamic audit for malicious behavior of mobile terminal mainly includes four parts: malicious behavior feature extraction, malicious behavior audit, malicious behavior interception, and audit report generation. The process of malicious behavior audit is shown in Fig. 2:

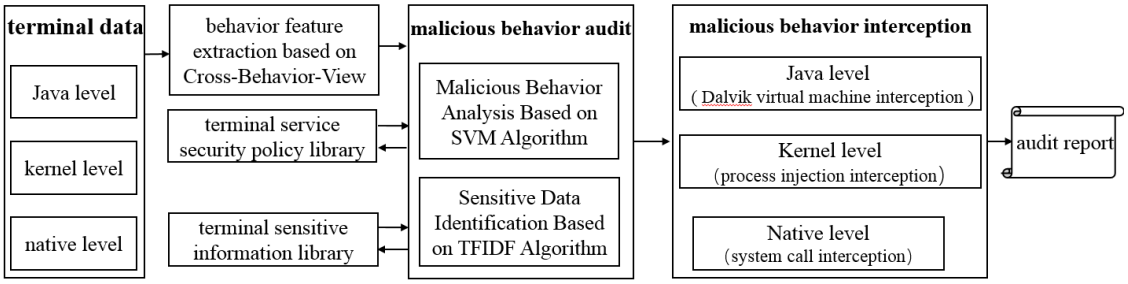

Figure 2. process of malicious behavior audit

Behavioral Characteristics Analysis. Software behavior information represents the current software operating state, which is used to distinguish whether current software is in a normal state or not. However, there are a great variety of information about software behavior, and each kind of

information indicates different state of the software. Therefore, it is necessary to extract the behavior information effectively and find the sensitive behavior which can directly reflect the software status and frequently occurring behavior. According to statistics, malicious behavior mainly acts as privacy stealing, tariff consumption, software backdoor, malicious fees and so on. The impact of these malicious mainly reflects on the mobile terminal about system information state, system resources and user information.

Multi-level Malicious Behavior Monitoring Model. This paper proposes a multi-level malicious behavior monitoring model from Java level, Native level, and Kernel level to achieve effective monitoring about malicious behavior of mobile terminal, so that we can extract the overall behavior of the audit features. Behavior monitoring on Java level adopts Dali virtual machine interception technology: it processes a Java method into a Native method at run-time, so that the virtual machine abandons the original execution sequence, in turn to perform additional pretended execution sequence. This method can intercept sensitive Java function of target application process or system process, to monitor malicious behavior at Java level.

Monitoring the malicious behavior at Native level mainly intercept and parse the Binder communication packet method based on process injection technology. Binder is an efficient and easy-to-use inter-process communication (IPC) mechanism. It is the Client / Server communication model, combined with the driver to promote efficiency of communication among several processes. Also, it improves performance through shared memory, while providing synchronization operation among processes. Interception of binder packet makes use of process injection technology to intercept IOCtl function, and replace with a user defined function. By analyzing the parameters of the function, that is, analysis of the binder communication packets we can monitor malicious behavior at Native level.

Monitoring the malicious behavior at Kernel level mainly hijack and replace system call method. System call is a set of call interface, which is provided by operating system for user program. The user program always obtains the necessary system services through the interface to the system kernel. In contrast with X86 architecture, Arm-Linux architecture adopts exception vector table to achieve system calls through the soft interruption swim instructions. The system call hijacking and replacement is based on the dynamic loadable module LKM mechanism, and specific operation of monitoring and recording is defined in the replaced function.

Behavioral Feature Extraction. In order to improve the identification accuracy of malicious behavior, the behavior data obtained from the multi-level monitoring model is selected according to the behavior feature selection rules. Also, behavior signature at Java level, Native level, Kernel level is respectively formed. Three-layer signatures are analyzed and compared to obtain hidden behavior characteristics. Specific principles and operation are as follows:

Firstly, define the basic entities in this module:

Pm={p|p is a malicious program with hidden behavior};

M={m|m is behavior monitoring procedure, referred to as monitor};

O={o|o is system's behaviors};

OM={o|o is visible behavior to monitoring program M};

OH={o|o is hidden behaviors}，according to the definition above, so OH=O−OM;

V=vb(m,O)，Where vb is the view generation function and m is the monitor.

Vt={v|v is trusted view};

Vu={v|v is non-trusted view};

$Vt \cap Vu = \emptyset$ and $Vt \cup Vu = V$;

VB={vb|vb is a function to generate view}，$vb: O \rightarrow Vo$;

CMP={cmp|cmp is a function to compare views}, $cmp: (v,v') \rightarrow OH$.

For monitor *m*, if $\exists Pm$ and the following two conditions are satisfied, then *Pm* is said to launch a hidden attack on the monitor *m*: If *Pm* is not running, and $\forall o \in O$, then $o \in OM$; if *Pm* is running, and $\exists o \in O$, then $o \notin OM$.

In this module, M={m_kernel,m_native,m_java}. m_kernel, m_native, m_java respectively stands for monitor at kernel, Native, Java level. Vt={v_kernel},v_kernel is behavior view generated

by m_kernel at Kernel level. Suppose v_kernel is trusted, then Vu={v_native,v_java}. Since underlying view confidence is higher than that of the upper view, v_native is more trustworthy than v_java. Because it still has the possibility of being bypassed by malicious programs, it is also an untrusted view.

Malicious Behavior Audit. Based on the multi-level monitoring of java level, native level and kernel level, the obtained monitoring data is transformed into the sample feature vector, and the SVM algorithm is used for learning. The data in the terminal business security policy can be taken as the training sample and brought into the SVM algorithm to construct the decision function as shown in equation 1.

$$y = f(X) = \mathrm{sgn}\left((W \times X) + b\right) \tag{1}$$

We classify the X vector and use the loss function $f(X)$ to obtain the parameters W and b of the hyper plane $W \times X + b = 0$ that minimize the error rate of the original sample classification, so as to establish the terminal malicious behavior discrimination model based on SVM algorithm.

Malicious Behavior Interception. The Business Security Threats Policy Library prefers a list of high-risk behaviors, as shown in Table 1, which collects eight categories of malware from the network, including malicious deductions, privacy theft, remote control, malware propagation, tariff depletion, system breaches, Rogue behavior, while covering send text messages, make phone calls, privacy stealing, remote control, elevation of authority and other malicious actions.

Table 1 List of high-risk behavior

Attributes	Related API
Device model	android.os.Build.MODEL
Compile tags	android.os.Build.TAGS
Telephone number	android.telephony.TelephonyManager.getLine1Number()
IMEI	android.telephony.TelephonyManager.getDeviceId()
IMSI	android.telephony.TelephonyManager.getSubscriberId()
Operator ID	android.telephony.TelephonyManager.getSimOperator()
Operator name	android.telephony.TelephonyManager.getSimOperatorName()
...	...

On the one hand, malicious behavior interception intercepts malicious behaviors preset in business security threat policy library. In order to not affect the normal operation of the program, we always intercept malicious behavior through the hook replaced with false data. On the other hand, the malicious behavior audit directly returns to the malicious application list which can be intercepted from the Java level, the Native level and the Kernel level. For example, when the malicious behavior occurs, the process will be interrupted instantly. In order to not affect the normal operation of the application, interception action is often achieved by returning false data. In addition, audit reports are generated based on the results of the behavioral audit.

Data Loss Prevention

The data loss prevention module of mobile terminal takes the data security as the core, combined with sensitive data automatic identification, sensitive data call blocking, data encryption and decryption, data desensitization, etc., The key technology of the whole life cycle security management for sensitive data is sensitive data identification. Since there is enough research on technology of data encryption and decryption [8, 9, , 10], data desensitization [11], we will not repeat them here.

The representation of the data is mainly based on the vector space model, which is described and calculated by representing the document as a vector. On this basis, a method of sensitive data recognition based on TFIDF algorithm is proposed based on the calculation of cosine similarity. The technical framework is shown in Fig. 3.

The data in the terminal sensitive information database is taken as a training sample and extracted into TFIDF algorithm. The feature of the known sensitive data is extracted and the data feature vector is used to calculate the cosine similarity to form the threshold. The sensitive data call blocks the blocking request from the receiving malicious behavior analysis detection module and blocks the process that the malicious behavior analysis detection module regards as maliciously calling the sensitive data, forcibly terminates its operation to prevent the loss of sensitive data, and alerts user and records. Sensitive data call blocking includes: (1) blocking the application permissions; (2) blocking the operation; (3) disable the application.

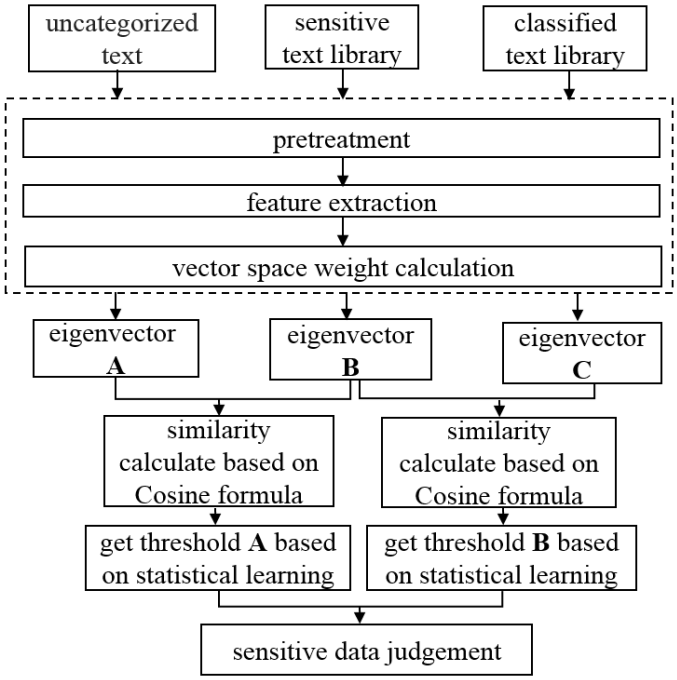

Figure 3. Framework for sensitive data recognition based on TFIDF algorithm

Conclusion

This paper presents a mobile terminal security monitoring system model based on distributed agent, which adopts distributed data acquisition and centralized management deployment. Aiming at the problem that malicious behavior analysis for mobile terminal cannot be detected hierarchically; a multi-level malicious behavior monitoring model is proposed to realize the hierarchical monitoring of mobile terminal behavior. In the future work, we will continue to improve the system performance, and deploy the system in the mobile information business for smart grid to ensure security for power mobile terminals in the information business.

References

[1] Isohara T,Takemori K and Kubota A.Kernel-based behavior analysis for android malware detection[C].7th International Conference on Computational Intelligence and Security,CIS 2011.IEEE Computer Society,2011:1011-1015.

[2] Berthome P.et al. Repackaging android applications for auditing access to private data[C].2012 Seventh International Conference on Availability, Reliability and Security (ARES).IEEE,2012:388-396.

[3] Guido M.et al. Automated identification of installed malicious Android applications[C].Digital Investigation.Elsevier Ltd,2013:96-104.
[4] Mitchell M,Tian G and Wang Z. Systematic audit of third-party Android phones[C].CODASPY 2014-Proceedings of the 4th ACM Conference on Data and Application Security and Privacy.Association for Computing Machinery,2014:175-186.
[5] Xiaochuan Miao, Security Analysis for Android Applications by Identifying Sensitive Routes[D], Nanjing University, 2016.
[6] Shabtai A, Kanonov U, Elovici Y, et al. "Andromaly": a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1): 161-190.
[7] Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution[C]//Proceedings of the 33rd IEEE Symposium on Security and Privacy (S & P'12). San Francisco, USA:IEEE, 2012:95-109.
[8] Goshwe N Y. Data encryption and decryption using RSA Algorithm in a Network Environment[J]. International Journal of Computer Science and Network Security (IJCSNS), 2013, 13(7): 9.
[9] Mathur A. A Research paper: An ASCII value based data encryption algorithm and its comparison with other symmetric data encryption algorithms[J]. International Journal on Computer Science and Engineering, 2012, 4(9): 1650.
[10] Zhou Y J, Jiang X X. Detecting passive content leaks and pollution in Android applications[C]//Proceedings of the 20th Network and Distributed System Security Symposium (NDSS'13). San Diego, USA:ISOC, 2013:1-16.
[11] Chin E, Felt A P, Greenwood K, et al. Analyzing inter-application communication in Android[C]//Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11). Washington D C, USA:ACM, 2011:239-252.