EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

An adaptive smartphone anomaly detection model based on data mining

Xue Li Hu*, Lian Cheng Zhang and Zhen Xing Wang

Abstract

With the popularization of smartphones, they have become the main target of malicious applications. In recent years, malware has become a major threat to Android smartphones. Detection for malicious applications in smartphone has become a research hotspot. There are many studies that were mainly based on host mode. Although this kind of approach has the advantage of an effective collection of client features, it can interfere with the original environment for detection and only applies to certain versions of phones. In this paper, we combine network traffic analysis with data mining to identify malicious network behavior. We improve Apriori algorithm to extract network traffic features from network data, and overall exposes malware functionalities through operational behavioral triggers. Then, it adopts a density-based local outlier factor (LOF) clustering algorithm to form a detection model. ADMDM model can effectively detect an anomaly, and it has good results in unknown anomaly detection. The proposed model can be used for daily smartphone security checking and evaluation. Moreover, ADMDM enriches techniques for dynamic smartphone behavior analysis.

Keywords: Anomaly detection, Apriori, Local outlier factor, Malware

1 Introduction

With the recent development of technology, smartphones have rapidly become an important part of our professional and personal life, thanks to their valuable features, such as excellent portability, great extensibility, and various functionalities. Moreover, the amount and value of information carried by smartphones have also increased dramatically along with the increased usage of phones. Mobile devices are becoming the main devices for web surfing. At the same time, more and more mobile phones have also become the targets of attackers. The consequences of smartphone attack are very severe. According to the 2017 annual network security report, the newly added Trojan virus accounts for 42.33% of the total number, and it is still the largest type of virus in smartphones. Malicious software represented by Trojans can steal user's privacy information or download and install other malicious software, which brings great harm to the user. Thus, it is better to detect anomaly as early as possible, in order to prevent possible losses. Several practical methods have been proposed. These malware detection methods can be divided into three categories: static analysis, dynamic system-level behavior analysis, and network-level behavior detection [1]. Although theoretical and practical achievements in smartphone malware analysis and detection have been made, there are still some limitations in them [2], such as insufficient smartness in an automatic dynamic analysis of behavioral triggers and lack in the Apps network behavior analysis.

Some researchers have begun to explore the use of network traffic to detect malicious applications [3]. Any version of smartphone based on any kind of operating system, as long as connected to the network, will produce network behavior. Since smartphone usage is mostly based on network traffic (3G, 4G, and WiFi), network traffic is a major carrier of network system information including recording and reflecting of network user activities. Therefore, we can identify malware by analyzing the critical behavior of application software in network traffic. A number of traffic features have been identified for malicious apps detection [3–8]. However, the prior arts have encountered some challenges, such as the lack of sufficient labeled traffic data, and most of the mobile devices intrusion detection systems use a host-based intrusion detection model [6–8], which means that the intrusion detection system is installed as

* Correspondence: success_receive@hotmail.com
State key laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China

a client on the smartphone. When the client collects, processes, and analyzes data, phone's original environment is changed, and the resource consumption of phone energy increases.

To address these challenges, we introduced a data mining-based model for adaptive anomaly detection (ADMDM), which detects smartphone anomaly by external network traffic examination. This approach is to capture external network behaviors, which will not have any impact on the detection of mobile phones, and it can be used on all Android versions. The principle of digital forensics has explicitly stated that when obtaining digital evidence, its originality shall not be changed. ADMDM has effectively guaranteed the primitiveness of the evidence and solved the key problem of the dynamic forensics of malicious behavior. ADMDM enriches techniques for dynamic smartphone behavior analysis, and it can meet smartphone users' security requirements.

The main contributions of this paper are as follows:

1. A lightweight smartphone adaptive anomaly detection model applicable to almost any version of Android is proposed.
2. Improved Apriori association rules can effectively mine malicious traffic features.
3. ADMDM does not rely on learning training datasets, which means that there are no limitations related to training dataset usage. Therefore, it provides good detection results for even unknown types of anomaly attacks.

2 Related research

At present, most researches are based on host's intrusion detection mode, which is through network traffic to detect Android malware mechanisms.

Jacoby et al. [6] described a battery usage-based intrusion detection system, which detects hidden intrusion behavior using the host intrusion detection engine (HIDE) that monitors the energy consumption. Energy consumption monitoring can detect irregular energy consumption behaviors and trigger other protection mechanisms through sensors. HIDE scanning port intrusion engine (SPIE) determines the IP and port resources of attacks. Host analysis signature tracking engine determines the attack energy signature and compares it to the most common attacks through the Apriori algorithm in order to provide protection and warnings between mobile host and networks.

Smart Siren [7] is an alarm system intended for smartphones with Windows Mobile 5.0 systems. It collects smartphone's network traffic information and sends it to the server, which uses statistics-based anomaly detection for malware detection, and if the server finds a malware, it sends warnings to the client, as well as to other mobile devices that interact with the client.

MONET [8] monitors running of smartphones with the Symbian system. It extracts the features of device status and uploads them to the remote server, which is responsible for anomaly detection analysis based on the uploaded feature data.

Crowdroid [9] is a lightweight client application, which collects applications' kernel system call sequences and uploads them to the server, wherein the frequency of system call is used as a feature and malicious codes are detected by k-nearest neighbors (KNN) algorithm. Crowdroid can detect malwares with identical names and version numbers, but with different malicious behaviors.

Danny et al. [10] introduced a lightweight approach for malware detection on the Android system. The proposed approach captures network packages and analyzes personal information exposure, as well as interactions between command and control servers. In the experiment, VirtualBox 4.0 × 86 simulator, which used × 86 compatible EeePC images running on Android 2.2 to capture network packages for personal information exposure analysis and interactions with the control server, was employed.

Moreover, Andromaly [11] is a host-based malicious behavior detection framework for Android. Andromaly monitors mobile devices for all kinds of features and events in real time. It collects monitored data, trains the anomaly detector with machine learning techniques, and labels the collected data as normal (good) or abnormal (malicious). During the experiments, the performance of Andromaly system was evaluated mainly by four malicious applications developed by researchers, which provided the features of malicious behaviors from limited samples.

Salman et al. [12] suggested a host modularization-based intrusion detection framework for mobile devices, DAIDS, which analyzes the applications on Android system by their behavior. DAIDS system monitors applications' behaviors in real time and supports dynamic analysis for abnormal and malicious behavior.

Shabtai et al. [13] proposed a C/S structured fused intrusion detection model. The client application monitors all running applications on the system and obtains their operational data for training and generation of the local user model, which has to detect differences by comparing application behavior to "normal" behavior. The collaboration-learning server is responsible for collecting and consolidating data reports from different mobile devices, from which it learns the collaboration patterns. This method is very effective for detecting self-updating malware but cannot cope with malicious applications that are obfuscation techniques such as encryption.

3 Methodology

In this paper, a novel data mining-based model for adaptive anomaly detection (ADMDM) is introduced. ADMDM firstly captures network traffic data, then performs IP linkage analysis on the weighted labels and uses Apriori algorithm for feature mining, and lastly, uses local outlier factor clustering algorithm to generate detection model adaptively, as shown in Fig. 1.

Although most of existing intrusion detection techniques classifies events as normal or abnormal, the definition for anomaly is not unified. Hawkins's definition of outliers is used in this paper, i.e., outliers are data that are the most far away from the other data in a dataset, which indicates that these biases are not randomly generated, but generated by a completely different mechanism.

In Fig. 1, the data capture step consists of smartphone's external network traffic data collection (data for examination) using data capture devices. Further, the data filtering module filters out noises and irrelevant data according to quantitative analysis. Subsequently, the IP linkage analysis module examines abnormal destination IP addresses. Then, traffic features are sent to the adaptive detecting module for generation of a detection model. Afterwards, the anomaly detection module determines whether the input data are normal or not and returns the detection results. In addition, the self-adaptive model captures data from the original system in order to generate an initial detection model, and when the change/update of network traffic features data reaches a certain level, the adaptive module will generate a new detection model through self-learning. Hence, automatic anomaly detection is achieved.

3.1 Data pre-processing

Massive raw data captured from the web are not immediately suitable for data mining. Data pre-processing should be done before the data mining process. Pre-processing includes data selection (selection of relevant data), data filtering (noises and redundant data removing), and data transformation (data conversion to the certain storage formats suitable for data mining). Through pre-processing, captured data are transformed into network traffic features, referred as traffic features, which represent the network behavior, upon which a clustering-based detection model is constructed.

At the pre-processing stage, raw data packages are converted into connection records. A connection record contains connection creation time, source IP address, source port, destination IP address, destination port, protocol, number of bytes in labels, number of data packages, uploaded data package amount, downloaded data package amount, and number of connections. These connection records can describe not only TCP and UDP flows but also HTTP flow.

ADMDM processes the connection records based on time window units, which means that network data are assigned into a series of time windows based on their occurring time and that the connection records are reconstructed from the raw network data within a time window. Moreover, TCP protocol defines that a data flow ends with the capture of RST data frame or FIN data frame. On the other hand, UDP protocol is a non-connection-based transferring protocol. Lastly, the data flow ends when no data frame is captured from either side of the communication channel within a certain period.

3.2 Destination IP linkage analysis

A destination IP indicates the destination of a connection. Captured IPs are labeled on the basis of their regions. Firstly, the belonging area of IP address is distinguished. Then, a query is executed in an anomaly IP address database. If a match is found, a warning of anomaly will be sent, and the anomaly IP address will be updated into the database in real time as well.

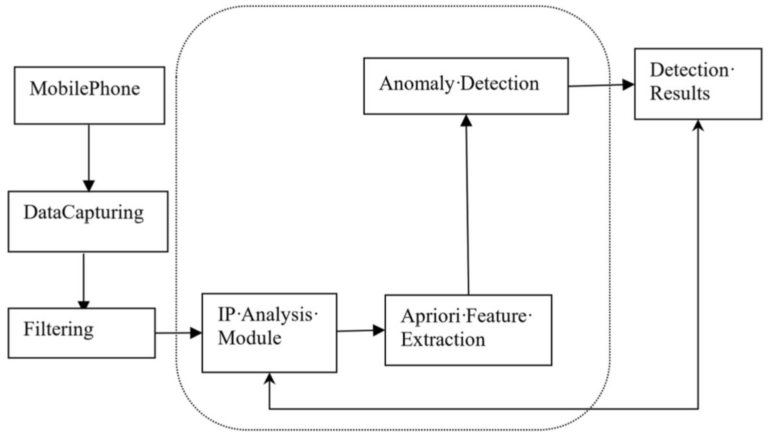

Fig. 1 ADMDM architecture. As shown in this figure, ADMDM firstly captures network traffic data, then performs IP linkage analysis on the weighted labels and uses Apriori algorithm for feature mining, and lastly, uses local outlier factor clustering algorithm to generate detection model adaptively

The blacklist effectively reduces the false negative rate, since some malware in order to hide the communication behavior, they used covert channel technology. The traditional network covert channel mainly uses the redundant fields of the transport layer and the network layer protocol to transmit information. However, due to the protocol firewall, the information constructed of the redundant fields can be easily detected. To meet certain convenience or to break the firewall barrier, modern network covert channel technology would encapsulate in a specific protocol format of steal data. The common protocols of covert tunnels include IRC protocol, DNS protocol, HTTP protocol, and so on.

3.3 Extract feature of network traffic

The traffic features are defined as the frequent item set with attribute composition, such as source address, destination address, service type, service protocol, duration, and connection status. And according to the network traffic including three stages which are establish the connection, keep the connection, interactive connection, we extracted different characteristics, a total of 19, as shown in Table 1.

3.3.1 Establish the connection

Malicious applications' successful implantation in the operation, then, make a request to DNS server to resolve the address IP, according to the DNS response, the address of server to connect the malware server terminal. When the IP address is valid and the server terminal is online, the connection is established successfully and accessed the next connection phase. When an IP address connection fails, the malicious application tries to iterate DNS requests intermittently and tries to connect until the connection is successful or offline.

Malicious applications and server communication run smoothly the establishment process of a connection only for requests and responses of DNS, which is not significantly different from the normal applications in DNS request process; the only difference is that the request domain name, if the domain is a malware that can be detected in the initial blacklist phase and be reported abnormal. Malicious applications are not successfully established and will produce differently from the normal website to access DNS requests.

In a short time, it will produce duplicate DNS requests, which stems from the design of the best-effort connection strategy. This strategy is very important that ensure malicious applications can rebound back to the server. In order to prevent malicious applications from making a large number of DNS requests caused by offline servers, some malicious applications adopt a silent mode of communication. Silent mode allows malicious applications join in non-communication mode in a short period and get better communication hidden.

Table 1 List of feature sets

Features names	Feature descriptions
Src_IP	Source IP
Dst_IP	Destination IP
Service_domains	Service domain name
Service	Service type (service port)
Protocol	Service protocol
Duration	Duration
Status	Status label for ending connection
Count_Src_Conn	Number of connections initiated from source
Count_Des_Conn	Number of connections initiated from destination
Count_Src_Data	Total data traffic amount sent from source
Count_Dest_Data	Total data traffic amount sent from destination
Count_Src_Retransmitted	Number of retransmitted connections from source
Count_Dest_Retransmitted	Number of retransmitted connections from destination
Src_Get_message	encrypted-1;non-encrypted-0
Src_Post_message	encrypted-1;non-encrypted-0
Fre_SrcIP	Activity frequency of the TCP data flow from source IP within one time window
Fre_DestIP	Activity frequency of the TCP data flow from destination IP within one time window
Count_conn	Number of connections with identical service, destination IP, and status label within a certain period of time
Count_total_conn	Number of total connections within one time window

This table included 19 features, such as source address, destination address, service type, service protocol, duration, connection status and so on. These features included three stages of network traffic, which were establish the connection, keep the connection, Interactive the connection

1. According the DNS anomaly in establishing connection, we extract the following features:
   The number of DNS requests in a certain period of time (unit: bar).
   In order to promptly on-line, malicious applications will produce a large number of DNS request packet in a short time.
2. According the silent mode of malicious applications, we extract the following features:
   After receiving the DNS response packet, the number of communication between IP in the request and response (unit: bar). In silent mode, Malicious application does not make any connection with the requested IP.

3.3.2 Keep the connection

When the connection is successful, the control server is not operating immediately, then malicious applications are waiting to keep the command. Heartbeat mechanism is a common strategy, which is used by malicious application, especially Trojan.

Of course, the heartbeat mechanism is not only used for malicious applications but also used often for normal applications. There are two ways to send a heartbeat packet, the first is to use Keep Alive mechanism of TCP, and the second is to implement own heartbeat packet in the application layer.

But in the entire communication process of a malicious application, "heartbeat package" does not exist alone, that accompanied by some confirmed data packets, the whole process is called the "heartbeat process", between the time interval of two adjacent "heartbeat process" is called "heartbeat gap." The continuous "heartbeat process" and "heartbeat gap" composed of alternating communication behavior is called "heartbeat".

According to the heartbeat mechanism, the following features can be extracted:

1. The packet ratio of receive and send of "heartbeat process"
   In each "heartbeat process," the malicious applications have the same behavior; thus, the proportion of packets sent and received is essentially constant.
2. The time length of "heartbeat gap"
   There is a big difference between the time length of "heartbeat gaps" for different malicious applications, but for the same sample, they have a fixed length of time. However, the "heartbeat gap" is not as constant as the "heartbeat process." To evade the statistical analysis, some malicious applications used a variable "heartbeat gap," which purpose is to hide the constant "heartbeat process" with changing "heartbeat gaps"; thus, it will be irregular. During normal network communication, the initiation of access is completely random. However, in order to ensure stable communication, the malicious applications are usually only pseudo-random "heartbeat gap" by using various algorithms; these behavior are still different from normal network communication.

3.3.3 Interactive connection

When the connection is successful, malicious applications and servers enter the interactive process. As a view of resource, the malicious applications as a conveyor are similar to the normal server role, and the normal resource request is the opposite.

1. Communication during transmission
   In the interactive connection phase, the malicious server sends a remote control command to the malicious application.
   The malicious application received command and executes the corresponding operation, then they return the result to the server. In this process, the target system will send a lot of resources.
2. The distribution of packet in communication
   In general, the design of the commands transmitted during the connection is based on the simple principle. A large of sample analyses show that the command length of most samples is less than 200 bytes. This means the size of the data packet that is sent by the malicious server is also small. In the communication process, malicious application not only return simple response information but also return a lot of data, the volume much more than the receive data package. Therefore, the size of the data packets sent by the malicious application terminal is usually large, and the number of data packets sent is larger than the number of data packets received.
   Data flow in communication.
   As the perspective of communication traffic, the malicious application purpose is to steal a lot of user data, so the amount of data uploaded is usually larger than the amount of data downloaded; the overall data flow is from the inside to the outside.
   Based on the research, we extract the following features:
   - The total number of session packets (total packets, unit: num).
   - Session duration (duration, unit: second).
   - Session packet number (small packets, unit: months). The definition of packet length less than 200 bytes, otherwise known as a large package.
   - Session to receive the number of packets/session packet number (Revc small packets/small

Packets, units: none). In sessions, malicious applications received are control commands, the received packets are small packets, and the information sent by the malicious applications is mainly large packets. So the value is generally greater than 0.5.

- Send large packets/large packets (unit: none). The value is generally greater than 0.5.
- Number of sessions received packets/number of sent packets (Revc packets/send packets, none). The value is generally greater than 1.
- Session upload data/download data (upload/ download, unit: none). The fact that during communication, the amount of uploaded data is larger than the amount of data downloaded; thus, the characteristic value is generally greater than 1.

3.4 Improve Apriori mining algorithm

ADMDM uses Apriori algorithm to convert captured connection records into traffic features. Apriori algorithm is a data mining technique, which represents the knowledge of dependency or correlation between a certain event and other events. Apriori analysis can help to discover the correlation between fields in recorded data. For instance, most of privacy theft Trojans contains a controlling side and a controlled side (Trojan side). The controlling side will send messages to the controlled side in real time in order to monitor the status of the controlled side. Heartbeat packages may have periodic behaviors or randomized behaviors, in order to avoid to be detected by the statistical analysis. Nevertheless, communication between the controlling side and the controlled side cannot be detected from a single TCP connection record. However, if we analyze similar TCP connections generated with the anomaly destination IP, within a certain period, these behaviors can be captured to detect Trojan.

Association rule mining represents the finding of all association rules in the given dataset *T*, whose support is equal or greater than the *min_support* and confidence is equal or greater than the *min_confidence*. Apriori association algorithm is a frequent item set algorithm for association rule mining. Its key concept is to mine frequent item sets through candidate set generation and event downwards closure examination.

In this paper, we use the timestamp aggregate conversation stream to get the sequence of clusters, every cluster corresponds to one transaction in the database, and for each item in the transaction corresponding to the packets in the cluster, thus we can use Apriori algorithm to mine the frequent patterns of packet. However, there is no sequence between the items in Apriori and there is no duplication. To apply the Apriori, some improvements need to be made.

Apriori algorithm is improved based on Apriori, GSP, and SPADE. According to the strict sequence of traffic, we added stronger frequent restrictions when frequent two-member sequences generate frequent three-member sequences or even more members.

1. The frequent three-item sequence must appear pairwise in pairs of frequent two-item sequences. Count the number of frequent of the element in the set, and exclude items that appear only once.
2. For the members P, A, and B, when P->A and P->B, if A->B exists, there may be exit P->A->B; When B->A exists, then may exist P->B->A. If A and B are irrelevant, then P->A and P->B cannot form frequent three-item, the candidate sequences can be excluded.

Similarly, there are rules to reduce computational complexity when frequent k terms are derived from frequent k-1 terms.

1. It is impossible for frequent k-items to be formed if an element in a frequent set of k-1 member sequences appears less than k-1 times.
2. The necessary condition for frequent k-items to be deduced from frequent k-1 items is that there is a sub-set in the set of frequent k-1 sequences, the sub-set contains k elements, and the number of occurrences of each element is equal for k-1 times.

The improved method gives a full play to the two main features of Apriori: k-1 items set connection rules and anti-monotonicity.

We can obtain some features of anomaly traffic through association rules. For instance, communication between the controlling side of the Trojan and the Trojan side can lead to appearance of the abnormal IP addresses and increase in upload network traffic. For anomaly detection based on network traffic features, parameter thresholds must be firstly determined. In this paper, local outlier factor clustering algorithm is used to determine thresholds.

3.5 Anomaly detection model

Clustering, also referred as clustering analysis, is an unsupervised learning procedure. Everitt defined clustering as follows: objects within one class or cluster are similar to each other, while objects from different classes or clusters are different. A class or a cluster is a group of points with high similarity gathered in the testing space. Distances between any two points within one cluster are shorter than that between any two points belonging to different clusters. Class or cluster can be described as a connected region with relatively high point density in a multidimensional space. Lastly, it distinguishes low-density regions from others by density differences.

Clustering is an effective approach to analyze and categorize unknown data types. Through it, dense and sparse regions, overall distribution patterns, and relationships between data attributes can be recognized. Clustering approaches include division clustering [14], hierarchical clustering [15], density clustering [16], grid clustering, etc. Each approach has its own strengths and weaknesses. In this paper, it is considered that 90% of smartphone network traffic is generally normal in real-world; and in order to effectively detect the 10% anomaly traffic, we decided to use a density-based clustering method, the local outlier factor (LOF) [17, 18].

The density-based LOF detection method does not view an anomaly point in a testing space as a binary attribute, i.e., normal or abnormal. Instead, it gives a quantitative estimation of the anomaly level (LOF) of the point, *p*. It can quite accurately detect anomaly points in uneven distributions. Greater LOF (*p*) corresponds to higher anomaly probability and vice versa.

LOF [19] work flow is as follows:

Denotations of LOF: *o*, *p*, and *q* denote the objects in the dataset; $d(p,q)$ defines the distance between *p* and.

q; $d(p,c)$ represents the minimum distance between point *p* and any point *q* in set *C*, i.e., $d(p, C) = \min\{d(p,q) \mid q\ (C)\}$.

– *k*-distance of object *p*

For any positive integer *k*, define the k^{th} distance of object *p* which is the distance between *p* and *o* as *k-distance*(*p*), where object *o* satisfies the following conditions:

- There are at least *k* objects $o' \in D\backslash\{p\}$ that satisfy the condition $d(p, o') \leq d(p, o)$,
- There are up to (*k*-1) objects $o' \in D\backslash\{p\}$ that satisfy the condition $d(p, o') < d(p, o)$.

– *k-distance* neighboring area of object *p* ($N_{\text{k-distance}}$)

If we assume that the *k-distance* of object *p* is known, the *k-distance* neighboring area of *p* contains all the objects whose distances from *p* are less than or equal to *k-distance*, i.e. $N_{\text{k-distance(p)}}(p) = \{q\ D\backslash\{p\} \mid d(p, q) \leq k\text{-}distance(p)\}$. Such object *q* is called *k-distance* neighbor of object *p*.

– Reach distance of object *p*

The distance from object *p* to object *o* is defined as: for any given integer *k*, reach distance from object *p* to object *o* is $reach\text{-}dist_k(p, o) = max\ \{k\text{-distance}(o), d(p, o)\}$.

– Local reachable density of object *p*
The local reachable density of object *p* is the invert of the average reach distance between object *p* and its *MinPts* neighboring area.

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o\in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right) \quad (1)$$

– Local outlier factor of object *p*

The LOF of object *p* represents its level of an anomaly, and it is defined as:

$$LOF_{MinPts}(p) = \frac{\sum_{o\in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|} \quad (2)$$

Greater LOF value corresponds to higher anomaly probability and vice versa. A point close to the cluster center, with a LOF value of about 1, should not be considered as a local outlier. On the other hand, points at cluster boundary or outside a cluster have higher LOF values, and their probability of being outliers is also higher.

4 Results and discussion

In this paper, three different versions of Android systems (4.2.1,4.4.2, 5.0.1) were used to verify ADMDM detection performances. Malware samples are from Drebin [1] virus sample database.

4.1 Experimental dataset

Malware samples used in the experiment were connection records generated by six types of top ten malwares on Android mobile phone platform: Plankton (data theft), Droidkungfu (data theft, fraudulent links, malicious download, malicious cracking), Geinmini (data theft, malicious download, spyware), Gappusin (malicious download, data theft, malicious advertisement), Ginmaster (malicious cracking, malicious download), and Fake (misused value-added services). Normal samples consisted of connection records (used as the baseline) generated by the original Android system (including manufactory pre-install application) and normal applications from Google Play store.

Clarification of experimental data:

1. Based on different harmful consequences caused by different malicious code families, in this paper, malicious attacks were categorized into four types: data theft, malicious download, remote control, and attack of service denial.
2. Frequently occurring data composition recorded from the original Android system and normal applications were considered as a normal dataset. It was assumed that if an anomaly behavior occurs, the features of anomaly dataset can be obtained by excluding normal data composition from the acquired frequent data composition during attacks.
3. A network connection was defined as the TCP data package sequence collected from the beginning to the end of a certain time period, and the network communication generated by sending data from source IP address to destination IP address under predefined protocols (e.g., TCP and UDP).
4. Two testing datasets were constructed to verify ADMDM detection performances.

The first dataset was anomalies in network connection records without any real operations. We selected 600 applications from the six malware families and installed them on three different versions of Android systems, respectively. Network traffic data were recorded from each device for 2 weeks and labeled as no-operation.

The second dataset was based on anomaly connection records with operations. Same as above, users were asked to use devices with malicious applications for 2 h every day. Network traffic data were recorded for 2 weeks and labeled as having-operation.

4.2 Anomaly communication detection

4.2.1 Association analysis

The result of the association analysis was a set of frequent item sets with the compositions of several attributes. These attribute compositions have a high probability of representing the features of some attacking data packages. For instance, we detected an attribute composition (IP = 58.63.244.72, protocol type = http, service requested = encrypt) through association analysis. This composition occurred with a certain frequency, occasionally, and this characteristic looks like Trojan communication data package. The results of improve Apriori association rule based mining are listed in Table 2.

We according to the packet attribute characteristics in Table 1, such as Src_IP,Dst_IP,Service_domains, Service, Protocol and so on, as the input of the Apriori algorithm. The result of the Apriori algorithm analysis is a combination of several attributes of frequent itemsets, as shown in Table 2.

4.2.2 Anomaly detection

The dataset used in the experiment contained 2-week network traffic data, 250,000 records in total, from which we selected 1000 normal connection records and 200 anomaly records (with two types of attacks) as the training dataset for detection model generation.

Furthermore, 8000 network connection records were randomly chosen and divided into eight testing subsets: test1, test2, test3, test4, test5, test6, test7, and test8. Each subset contained 1000 network traffic records, of which 90% were normal behaviors and 10% were anomaly behaviors. Datasets test1–test4 labeled as no-operation contained four types of attacks and datasets test5–test8 labeled as having-operation contained four types of attacks.

We performed anomaly detection tests on eight datasets, using three different algorithms, LOF, KNN [20], and SVM [21], and the comparison results are shown in Figs. 2 and 3. In this paper, detection rate greater than 50% was considered as a successful detection.

In order to evaluate the detection results for the anomaly detection models effectively, the TPR [22] and FPR were selected as Evaluation index. In the following equation, the sensitivity and specificity are denoted as TRP and FPR, respectively.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \quad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$$

- TPR: the sensitivity reflects the coverage degree of a positive class. The definition of true positive rate is the ratio of number of positive samples correctly identified to all positive samples in the test set.

Table 2 3-itemset

Item	Support level
Dst_IP, Count_Src_Data, Src_ encrypt_data	628
Dst_IP, Count_Dest_Data, Duration	596
Status, Count_total_conn, Fre_DestIP	553
Src_IP, Count_Src_Conn, Status	545
Dst_IP, Protocol, encrypt_data	436
Fre_DestIP, Count_Src_Data, Count_total_conn	378
Src_IP, Fre_SrcIP, Count_same_dest	237
Src_IP, Status, Count_Src_Retransmitted	137
Status, Dst_IP, Count_Src_Retransmitted	135
Dst_IP, Count_Dest_Retransmitted, Count_conn	118

This table shows the result of the Apriori algorithm analysis

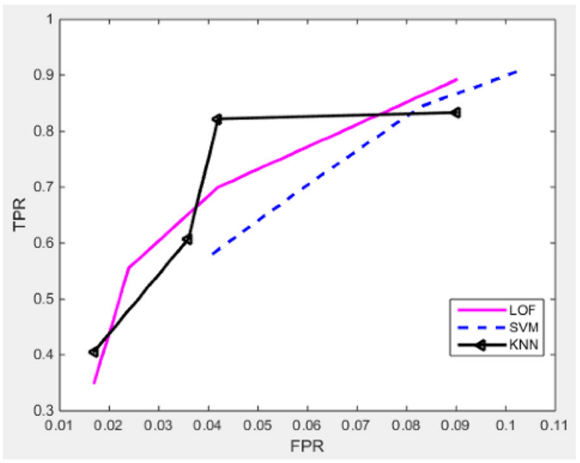

Fig. 2 The result of test1–test4. This figure shows the comparison results, which performed anomaly detection tests, using three different algorithms, LOF, KNN, and SVM on test1–test 4 datasets

- FPR: specificity presents a false positive rate. The definition of false positive rate is the ratio of the number of negative samples were identified as positive samples to all negative samples in the test set.

From detection results presented in Figs. 2 and 3, it can be noticed that unsupervised SVM algorithm has achieved good detection results on both types of datasets, test1–test4 and test5–test8, respectively. However, the high detection rate of SVM is based on keeping the false alarm rate to be above 4%. On the other hand, LOF and KNN have better detection performance when the false alarm rate is below 4%.

The results of test1–test4 show that KNN has better performance than LOF. The main reason for this is that when there is no operation triggering the malwares after successful installation, most of the malwares will be in idling state. LOF needs some regular anomaly connection occurrences of communications with the server, and these anomalies are usually in the sparse region of normal data, so LOF cannot detect anomaly with very low local density, whereas KNN can recognize them as outliers since their distances to the closest cluster are still very long.

The results for test5–test8 which were labeled as having-operations show that LOF has better performance than KNN. To hide well, the anomaly will generate a complete connection in most of the cases. These complete attack connections can be nicely mixed in the normal connection region, i.e., the density of anomaly is dependent on that of normal behaviors. Due to that, detections become more complicated when there are anomaly behaviors, such as malicious downloading or uploading of private information. LOF has the advantage of dealing with complicated situations. Namely, it performs detection based on local density instead of global density. It estimates normal points as outliers correctly due to different disperse conditions of data densities. Therefore, it is more suitable than KNN to detect an anomaly in complicated cases. KNN cannot detect this type of anomaly since it only calculates simple distances between sample points.

After the result analysis, we noticed that LOF algorithm has better detection performance, especially in complicated cases. It can maintain a detection rate of about 80% with a false alarm rate kept under 4%.

To further verify the ability of ADMDM to detect unknown samples, we had made four malware samples and collected 18 malware samples from the Internet, a total of 22 unknown samples which were installed in three different versions of Android smartphone. The results are shown in Table 3.

The detection performance of ADMDM is better on versions older than 4.2.1 with version 4.4 and 5.0. In 4.2.1 version of mobile phone detection, 18 malware samples were detected 14, while the laboratory 4 samples were completely detected, the overall detection rate of 81.8%. In 4.4 version of the detection rate of 72.7%, in 5.0.1 version of the detection rate of only 54.5%.

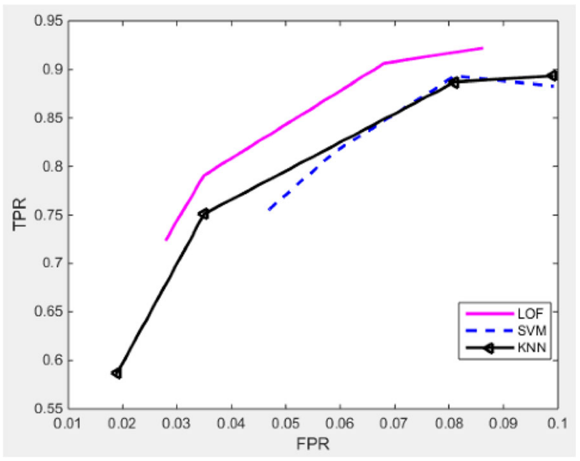

Fig. 3 The result of test5–test8. This figure shows the comparison results, which performed anomaly detection tests, using three different algorithms, LOF, KNN, and SVM on test5–test 8 datasets

Table 3 Detection results of unknown sample

N\Ver	Android 4.2.1	Android 4.4.2	Android 5.0.1
18	14	12	8
4	4	4	4
22	18	16	12
Detection (%)	81.80	72.70	54.50

This table shows the results of ADMDM to detect unknown samples. The unknown samples come from the laboratory and Internet

The reason for the low detection rate is that hardware compatibility problems, the Internet collected samples are relatively old that cannot be installed and allowed to run separately, and many samples of the server has stopped providing services. But laboratory malware samples can be based on mobile version and hardware requirements to do the appropriate changes, it can be installed on different versions of the phone to run and also be ADMDM detection, The other malicious samples cannot work regularly and produce communication flow, It cannot be detected thus reducing the detection rate. The results show that ADMDM has a good detection effect on unknown samples.

5 Conclusions

In this paper, ADMDM, a novel research which uses data mining to detect features of smartphone malwares traffic data, is presented. The proposed approach can obtain the features of anomaly network traffic without interfering with the original state of the device, which maximizes the protection of the forensic environment. The detection model can be used for smartphone security evaluation, which is very suitable for real-life applications. ADMDM uses Apriori algorithm to extract the features from the obtained network traffic, which effectively reduces the data amount for subsequent clustering algorithms. Moreover, local outlier factor (LOF) algorithm uses the concept of the local outliers instead of the conventional concept of binary division on outliers and non-outliers which was used in all of the previous outlier definitions. Furthermore, ADMDM can detect local anomaly with better accuracy and performance. Nevertheless, although the ADMDM can automatically and effectively detect smartphone network traffic anomaly, there are still some limitations in detection of the deeply hidden anomaly, which will be studied in our future works.

Abbreviations

ADMDM: Anomaly Detection Model based on Data Mining; KNN: k-nearest neighbor; LOF: Local outlier factor; SVM: Support vector machine

Acknowledgements

The research presented in this paper was supported by Ministry of Science and Technology, Taiwan.

Funding

The authors acknowledge the National Natural Science Foundation of China (Grant: 61402526).

Authors' contributions

XLH is the main writer of this paper. She proposed the main idea, deduced the performance of ADMDM detection, completed the simulation, and analyzed the result. LCZ introduced the LOF-based algorithm in anomaly detection. ZXW gave some important suggestions for the paper. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 26 February 2018 Accepted: 30 May 2018
Published online: 14 June 2018

References

1. D Arp, M Spreitzenbarth, M Hubner, H Gascon, K Rieck, *Drebin: effective and explainable detection of android malware in your pocket* (Network & Distributed System Security Symposium (NDSS) Conference, 2014), p. 23247
2. G Peng, J Li, R Sun, Y Xiao, Android malware detection research and development. J. Wuhan Univ., **61**, 021–033 (2015)
3. A Feizollah, NB Anuar, R Salleh, AWA Wahab, A review on feature selection in mobile malware detection. Digit. Invest. **13**, 22–37 (2015)
4. M Conti, LV Mancini, R Spolaor, NV Verde, Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Inf. Forensics Secur. **11**(1), 114–125 (2016)
5. H Zhang, D Yao, N Ramakrishnan, Z Zhang, Causality reasoning about network events for detecting stealthy malware activities. Comput. Secur. **58**, 180–198 (2016)
6. GA Jacoby, R Marchany, NJI Davis, Battery-based intrusion detection a first line of defense. Inf. Assur. Workshop **3**(9), 272–279 (2004)
7. J Cheng, SHY Wong, H Yang, S Lu, *Smartsiren: virus detection and alert for smartphones* (Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, 2007), pp. 258–271
8. AD Schmidt, F Peters, F Lamour, C Scheel, SA Camtepe, S Albayrak, Monitoring smartphones for anomaly detection. Mob. Netw. Appl. **14**(1), 92–106 (2009)
9. I Burguera, U Zurutuza, S Nadjm-Tehrani, in *Acm Workshop on Security & Privacy in Smartphones & Mobile Devices*. Crowdroid: behavior-based malware detection system for android (2011), pp. 15–26
10. D Iland, A Pucher, T Sch, *Detecting Android Malware on Network Level University of California* (2011), pp. 1–7
11. A Shabtai, U Kanonov, Y Elovici, "Andromaly": a behavioral malware detection framework for android devices. J. Intell. Inf. Syst. **38**(1), 161–190 (2012)
12. A Salman, IH Elhajj, A Chehab, A Kayssi, in *International Conference on Advanced Information Networking & Applications Workshops*. DAIDS: an architecture for modular mobile IDS (2014), pp. 328–333
13. A Shabtai, L Tenenboim-Chekina, D Mimran, L Rokach, B Shapira, Y Elovici, Mobile malware detection through analysis of deviations in application network behavior. Comput. Secur. **43**(6), 1–18 (2014)
14. Y Mo, Z Zhang, W Meng, M Lin, Y Wang, A spatial division clustering method and low dimensional feature extraction technique based indoor positioning system. J. Sens. **14**(1), 1850–1876 (2014)
15. F Murtagh, P Legendre, Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion? J. Classif. **31**, 274–295 (2014)
16. D Xu, Y Tian, A comprehensive survey of clustering algorithms. Ann. Data. Sci. **2**(2), 165–193 (2015)
17. M Radovanovic, A Nanopoulos, M Ivanovic, Reverse nearest neighbors in unsupervised. IEEE Trans. Knowl. Data Eng. **27**(5), 1369–1382 (2015)
18. GO Campos, A Zimek, J Sander, ME Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min. Knowl. Disc. **30**, 891–927 (2016)
19. M Breunig, HP Kriegel, R Ng, et al., in *Proc. of ACM SIGMOD Conference, Dallas*. LOF: dentifying density-based local outliers (2000), pp. 93–104
20. Z Deng, X Zhun, D Cheng, M Zong, S Zhang, Efficient kNN classification algorithm for big data. J. Neurocomput. **195**, 143–148 (2016)
21. VS Bin Gu, A Sheng, Robust regularization path algorithm for ν-support vector classification. IEEE Trans. Neural Netw. Learn. Syst. **28**(5), 1241–1248 (2017)
22. XL Hu, S Yi, ZX Wang, LC Zhang, Android Malware detection method based on system call sequence feature weighting. Acta Tech. **62**(3B), 371–380 (2017)