Chatting Application Monitoring on Android System and its Detection based on the Correlation Test

Yafei Li *, Jiageng Chen* and Anthony TS Ho †
* Central China Normal University, Wuhan, China
E-mail: chinakako@gmail.com
† University of Surrey, Guildford, UK
E-mail: a.ho@surrey.ac.uk

***Abstract*—Mobile phones are playing an important roles in our modern digital society, which have already replaced the traditional computer in many situations. Nevertheless, the number of malicious software also starts to grow and showed significant impact on our legal use. Among several mobile systems, the Android platform is currently the most widely used and open system, which also makes it a very attractive target for the malicious applications. User privacy is of great interest to many different agents, which becomes of the most valuable target for the malware, and the chatting software naturally become one of the richest information resource target. In this paper, we first investigate the core techniques that are used by the most monitoring softwares. Then we propose several correlation experiments to efficiently detect the those softwares. We developed a monitoring prototype as well as the detecting system, including the mobile phone side and the remote web server side, to simulate the scenario in the real-world environment. The experiment confirmed the efficiency of our approach.**

***Keywords*—Spy Applications, Android System, malicious software, Detection, Correlation Test.**

I. Introduction

In recent years, with the development of technology and the improvement of people's living standard, smart phone has become an important part of mass consumption and life. The functionality of smart phones is also becoming more and more powerful with the development of the technology, and it has somehow replaced other functional devices, such as the cameras, music players, and even as a productivity tool to deal with the tasks of people's life and work. Therefore, these factors lead to the fact that smart phones store a lot of user privacy data. Especially in recent years, with the rapid development of mobile Internet and cloud computing technology, users' on-line chat, mobile office and payment have become daily behaviors.

The Android operation system was introduced by Google company in 2007. Because of its open source characteristics, it soon became the most popular operation system in the market. According to IDC statistics, as of the first quarter of 2017, the Android market share has reached 85% in the global smartphone market. Such a high market share has attracted a large number of application developers from all over the world to develop various applications on their platform. But the Google play lacks a strict and meticulous procedure for testing the application market, and the mobile phone manufacturers provide their own mobile application stores, such as the Samsung application store, the HUAWEI application market and so on, which make the Android platform a breeding ground for the spread of the virus. At the same time, in order to reflect the difference of their products, mobile vendors have made a lot of customization to the native Android system, which also caused the fragmentation of the Android environment, and the manufacturer's defect repair and upgrading of the system is also behind the official system of the Android.

User privacy becomes the main target of the malware, and the chatting software due to the rich information it can generate, naturally become one of the most valuable target. The most widely used chatting softwares include wechat, QQ, whatsapp, line and so on [25], [26], [27], [28]. Most of the modern chatting softwares take advantage of the modern cryptography to achieve confidentiality, integrity and so on. Applications like whatsapp can even achieve end-to-end security [29]. Thus even end-to-end security is not available, one cannot easily obtain the message content by just passively intercept the wireless digital communication. As a result the attacker has to make the spy activity happen before the data is encrypted. At present, there are commercialized spy software in the market, such as mSpy, Highster Mobile, FlexiSPY, iKeyMonitor, and PhoneSheriff [1]. Usually the software will be installed on the target devices after the users pay for the service. Users can spy on the phone remotely through a web-based interface. The monitoring targets include: SMS, call records, Internet browsing records, chat records in social software, etc.

In this paper, we investigate the monitoring techniques on the android platform and develop an efficient monitoring system which can be used to capture chatting data from wechat, QQ and line. Then we provide the test framework which can be used to detect the spying behavior of the monitoring software. We structured the paper as follows. Section II reviews related literature. We designed a real-time chat record system, named Chatspy, based on the Accessibility API in Section III, and the experiment setup and findings are described in Section IV. We conclude the paper in Section V.

II. Related work

N Peiravian and X Zhu [15] proposed to combine permission and API calls and use machine learning methods to detect malicious Android applications. Lker Burguera [6] capitalized on earlier approaches for dynamic analysis of application behavior for detecting malicious software. Asaf Shabtai and Uri Kanonov [8] evaluated several combinations of anomaly detection algorithms, and the results suggested that the proposed framework was effective in detecting malicious software on Android. AD Schmidt and R Bye [12] performed a statical analysis on the executables to extract their function calls in Android environment and they also presented a collaborative malicious software detection approach to extend these results. T Isohara and K Takemori [13] proposed a kernel-base behavior analysis for android malicious software inspection, and the result showed that their system could effectively detect malicious behaviors of the unknown applications. B Amos and H Turner [17] presented a STREAM framework, which was developed to enable rapid large-scale validation of mobile malicious software machine learning classifiers. KO Elish and X Shu [19] described a highly accurate classification approach for detecting malicious Android applications with better efficiency.

Due to the development of machine learning and deep learning in recent years, more and more works are done to identify malware based on the algorithms. In particular, Bayesian classifier plays a very important role in identifying these malware[11][18]. Yousra Aafer and Wenliang Du [10] have conducted a thorough analysis to extract relevant features to malicious software behavior and their results showed that they were able to achieve a high accuracy using kNN classifier. J Sahs and L Khan [14] presented a machine learning-based system which extracted a number of features and trained a One-Class Support Vector Machine in an offline manner for detecting malicious software on Android. H Gascon and F Yamaguchi [16] proposed a method for malicious software detection based on efficient embedding of function call graphs with an explicit feature map inspired by a linear-time graph kernel.

This work instead focuses especially on the spy softwares which only target the popular chatting softwares such as QQ, wechat, whatsapp and line. Our detection methodology thus specifically works for such kind of malicious softwares.

III. REAL-TIME OBTAIN CHAT RECORD SYSTEM BASED ON ANDROID ACCESSIBILITY

Considering that some people have difficulty in operating the android system, or obtaining the information from the system due to the visual, physical, or age constraints, Android provides accessibility features and serves to help users better use Android devices. Accessibility is generally referred to as barrier free or disabled in China. The official summary of Accessibility can be found in [21]. We will mainly use this functionality to build our monitoring application.

A. Technical principle

Accessibility related services and interfaces had been added for completing the auxiliary function in the Android 1.6 period, with AccessibilityService component as the entry, combined with key classes such as AccessibilityEvent, AccessibilityNodeInfo. AccessibilityService is an abstract class that inherits Service, and which is a system component, so it is different from the general Service, The use-method is as follows:

- Add permission: android.permission.BIND_ACCESSIBILITY_SERVICE.
- Add action:android.accessibilityservice.AccessibilityService.
- Provide a meta-data named android.accessibilityservice and provide XML as a configuration file for AccessibilityService, and the configuration file declares that the service receive event type, feedback type, and so on.

B. Real-time access chat record system ChatSpy

Based on the Accessibility API (Application Programming Interface), we firstly designed our spy application prototype ChatSpy which includes two sides, the mobile phone side and the remote web server side. It is able to directly obtain the real-time chat records in social applications without rooting the phone. After capturing the message, it then send the chat records to a designated remote Web server, which runs the receiving services in the background, so it is transparent-totheuser.

1) Development environment: We show our development environment as follows:

a) Mobile phone side: The development environment of the mobile phone side is shown in Table I.

TABLE I: Development environment of Mobile phone side

Development Tool	Android Version	Development Language
Android Studio 2.3.3	4.4.2	Java8

b) Remote Web server side: The development environment of the remote Web Server is shown in Table II.

2) System running process: After the system is initiated, the mobile phone (client) side is sending or receiving messages from the remote Web server side. The overall process is shown in Figure 1.

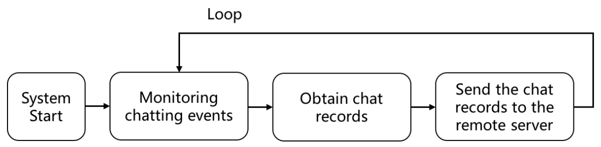
Loop
System Start
Monitoring chatting events
Obtain chat records
Send the chat records to the remote server

Fig. 1: System overall flow chart

After the mobile phone side application starts running, the Accessibility function is activated, which will receive the Accessibility event. Through the comprehensive processing of the package names (such as com.tencent.mm,

TABLE II: Development environment of Remote Web server side

Type	Description
Development Tool	Eclipse 4.6.2
Web Server	Tomcat 7
Development Language	Java 8
Database	MySQL 5.7.17
Dependency software package	Maven 4.0.0, Spring 4.3.8 SpringMVC 4.3.8, Mybatis 3.2.8 druid 1.0.31, slf4j-log4j12 1.6.4 tomcat7-maven-plugin 2.2, jsp-api 2.0 servlet-api 2.5, jstl 1.2 commons-net 3.3, commons-io 1.3.2 commons-lang3 3.3.2, junit 4.12 mysql-connector-java 5.1.32

com.tencent.mobileqq, com.whatsapp, jp.naver.line.android, etc.), text information, types of the events, and the acquisition of the real-time chat content will be realized. The flow chart is shown in Figure 2. After the remote Web server module is started, it monitors and receives the Http requests sent by the mobile phone, extracts the chat content from the http package, and persists the data to the database. The flow chart is shown in Figure 3.

3) System architecture: In terms of the system architecture, our ChatSpy is divided into two sides: Mobile phone client side and remote Web server side. The two parts work together to complete the real-time acquisition, transmission, data processing and storage of social applications' chat content.

a) Mobile phone side: The mobile phone side is made up of three parts: configuration files, Activity and function module. The detailed architecture is shown in Figure 4. The description of each functional modules and files in the project is shown in Table III.

Due to the different structure of each chat application, in terms of the monitoring events, we need to use different API to obtain the corresponding information. For example, to monitor the corresponding event we use TYPE_WINDOW_CONTENT_CHANGED for QQ and TYPE_VIEW_SCROLLED for Wechat, Whatsapp and Line. Table IV summarizes all the details we need for monitoring the different applications.

b) Remote Web server side: The remote Web server side uses the common Java Web programming, introduces the SSM framework (Spring, SpringMVC, Mybatis), and handles the chat content data obtained by listening to the http request from the client side, and stores the data by the data persistence layer. The architecture diagram is shown in Figure 5. The Web Server side mainly consists of three modules: Functional logic, Configuration and Database, as shown in Table V.

For example, one user receives a WeChat message "Hello, Bob" from Alice at 8:54 am, October 1st, 2018. ChatSpy will instantaneously obtain this message and send a http request: *http://ip/add/201810010854/WeChat: Alice send a message: Hello, Bob*. The remote web server will receive the request and parse the user name and the chat content, then store the message into the server side database.

4) System operation and test:

a) System running environment: We run our client side application on Huawei Mate 7 with Android 4.4.2 operation system, and installed four chatting softwares: Wechat, QQ, Whatsapp and Line. We use OSX system on the server side. Please refer to Table VI and VII for the detailed parameters.

b) System Test: ChatSpy does not have an user interface, which shares the similarity with a malware which usually runs in the background. Moreover, after starting the ChatSpy, we run the malware detection system which is the default application shipped with Huawei Mate 7, and to our surprise we pass the detection without any warning information. During the chatting procedure, we use the Accessibility API to catch the content of the screen and extract the corresponding messages we are interested in and send back to the remote server.

By testing our monitor prototype, we found that the Chatspy, designed by using Accessibility API, can obtain the user chatting records (wechat, QQ, whatsapp, line) in real time without rooting the device. In the next Section, we will focus on how to design an efficient strategy to detect the spy application which mainly target the chatting applications.

IV. EXPERIMENTAL PROCESS AND RESULTS ANALYSIS

Take the behavior of the spy application into consideration, it is reasonable to assume that there is a correlation between the activity of the chatting application and the spy application. Especially, the captured data needs to be sent to the remote backend server, which will trigger the network communication.

To obtain the test data, we design a module named GetNewWorkData that can get the instantaneous traffic of the running process. The application runs in the form of service in the background, monitors the instantaneous flow of the whole mobile system in real-time, and stores it in a txt file to facilitate data analysis in the future.

A. Data acquisition

ChatSpy detects that when the user receives or sends a chat record, it instantly encapsulates the chat record into packets and sends it to the remote server. Meanwhile, there exists many other running processes in the system such as the web browser and so on. From the detector's point of view, we need to take all these processes into consideration when perform the data analysis. For example, at 194s chatspy send and receive in total 290 bytes packet. Table VIII provides an example of several running processes. The other running processes remain in a rather inactive phase so we omit them here.

B. Data analysis

In order to test whether the current mobile environment has a malicious process running, we can analyze the network traffic values between two or more processes and the time interval generated by network traffic to determine whether these processes have a certain correlation. If a process has

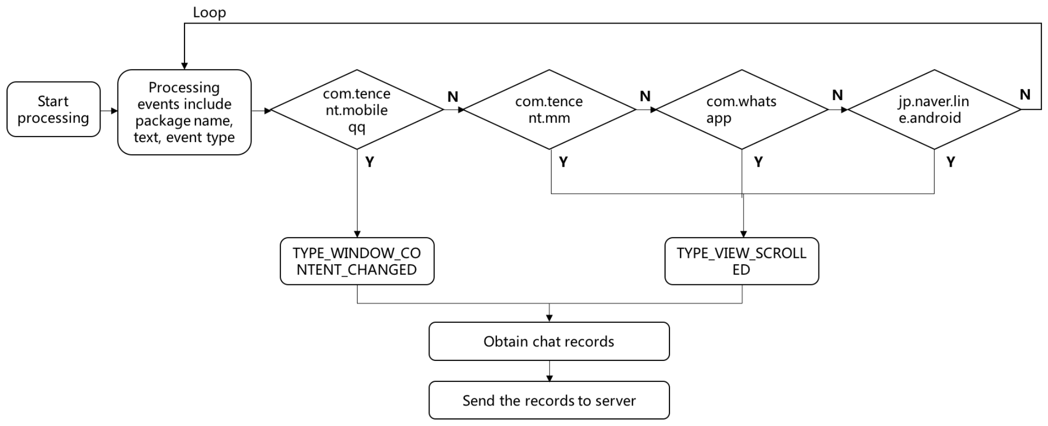

Fig. 2: Running process of the client side

TABLE III: Subset of the most relevant modules and files in Mobile phone side project

Module	File	Description
Configuration file	AndroidMainfest.xml	An entry file for the Android application,which describes the exposed components in package (activities, services, and so on) can be processed.
	ChatLogServiceConfig.xml	The main file to enable Accessibility services.It describes the types of events that need to be monitored, and the software packages that need to be monitored.
Activity	MainActivity.java	The core class of Android (android.app.Activity). In the Activity class, there is a onCreate event method, which is generally used to initialize the Activity, and the View is placed on the Activity via the setContentView method. After binding, the Activity displays the controls on the View.
Functional modules	ChatLogService.java QQChatLog.java WeChatLog.java WhatsappChatLog.java LineChatLog.java WebUtil.java	Get the chat record service Get the QQ chat record Get the WeChat chat record Get the Whatapp chat record Get the Line chat record Process chat record

TABLE IV: Different processing methods for each Applications

Social Application	Monitoring Event	How to get chat record	How to parse Username
QQ	TYPE_WINDOW_CONTENT_CHANGED	Judge the types of child nodes of chat nodes	By Control ID, we get the node set corresponding to the control, locate the last node, and resolve the name / nickname.
WeChat	TYPE_VIEW_SCROLLED	By Controll ID	
Whatsapp			
Line			

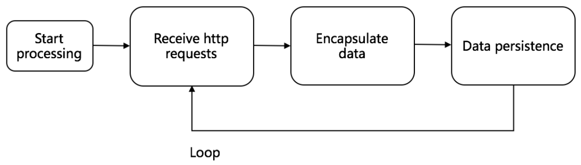

Fig. 3: Remote Web Server side running process

a significant correlation with the process of the social applications, it can be explained to a certain extent that the process is stealing chat records from the social applications.

1) Correlation analysis: Correlation analysis refers to the analysis of two or more correlated variables, so as to measure the relative degree of two variables. If the correlation between elements is related to some degree, the probability can be used to measure the evidence. he scope and areas of relevance cover almost all the aspects we have seen, and the definition of relevance varies greatly in different disciplines.

The common methods of correlation analysis include chart correlation analysis (line chart and scatter plot), covariance and covariance matrix, correlation coefficient, one element regression, multiple regression, information entropy and mutual information. We focus on the chart correlation analysis and the correlation coefficient in this paper.

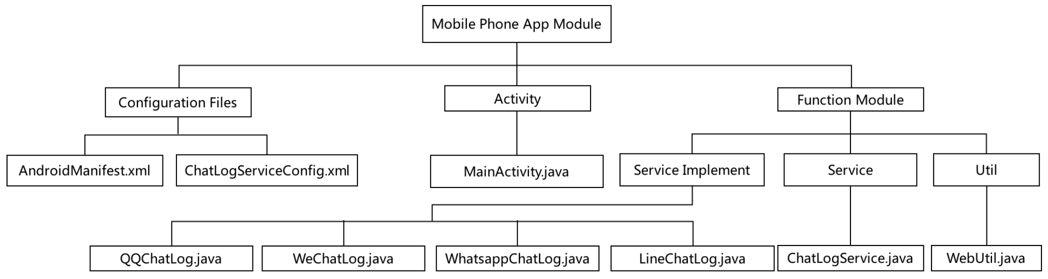

Fig. 4: Mobile phone side architecture

TABLE V: Subset of the most relevant modules and files in Web Server side project

Module	File	Description
Functional logic	MsgController.java	Located in the Controller layer, which is mainly responsible for receiving the request sent by the mobile terminal application, and then sending it to the service layer for processing.
	MsgService.java	Located on the Service level, and which is the interface to invoke the main business logic.
	MsgServiceImpl.java	Located on the Service level, which is the specific implementation of calling the interface of main business logic.
	MsgMapper.java	Located in the Persistence layer, and the main function is to map the entities in the chat record and the table fields in the database.
	Msg.java	Located in the Persistence layer, which is a chat record entity class.
Configuration	SqlMapConfig.xml	Configure the plug-in for mybatis.
	ApplicationContext-dao.xml	The management of the database by spring mainly includes configuring the database connection pool, loading the database configuration file, configuring SqlsessionFactory, configuring the scanning package, and loading the mapper proxy objects.
	ApplicationContext-service.xml	Spring configures the scan package and load the Service implementation class.
	ApplicationContext-trans.xml	Spring manages transactions, mainly including configuration transaction manager, transaction notification, and transaction section.
	Springmvc.xml	Configure the springmvc context, open the MVC annotation driver and configure the view resolver.
	Db.properties	Database configuration file.
	Log4j.properties	The log management configuration file.
	Web.xml	the configuration files of the whole Web project mainly include loading spring container, configuring filter to solve post garbled code, loading springmvc front-end controller, etc.
	Pom.xml	Maven configuration file.
Database	Messages	id varchar, which is used to store time nodes and the format is yyyyMMddHHmmssSSS. msg Longtext, which is used to store user chat text records.

TABLE VI: Running environment of Mobile phone side

Mobile phone	Operating System	Applications
HUAWEI Mate 7	Android 4.4.2	WeChat 6.5.4 QQ 7.2.0 Whatsapp 2.17.350 Line 7.3.0

TABLE VII: Running environment of Web Server side

Operating System	WebServer	Database
macOS High Sierra 10.13.3	Tomcat 7	MySQL 5.7.17

2) Correlation coefficient: The correlation coefficient is the first statistical index designed by the statistician Karl Pearson. It is the amount of the linear correlation between the

TABLE VIII: The network traffic of demonstrated processes

time(s)	ChatSpy	Browser	Google search	Chrome	Wechat	QQ
...	...	...	...	...	...	...
190	0	2921	1471	0	0	0
191	0	5816	0	0	0	0
192	0	7531	0	0	0	0
193	0	464	5596	0	0	0
194	290	60153	4456	0	0	595
195	290	73438	0	0	0	52
196	290	80941	0	0	0	3705
197	0	37657	1455	0	0	0
198	0	57610	0	0	0	0
199	0	34285	1507	0	0	0
200	0	78065	0	0	0	0
...	...	...	...	...	...	...

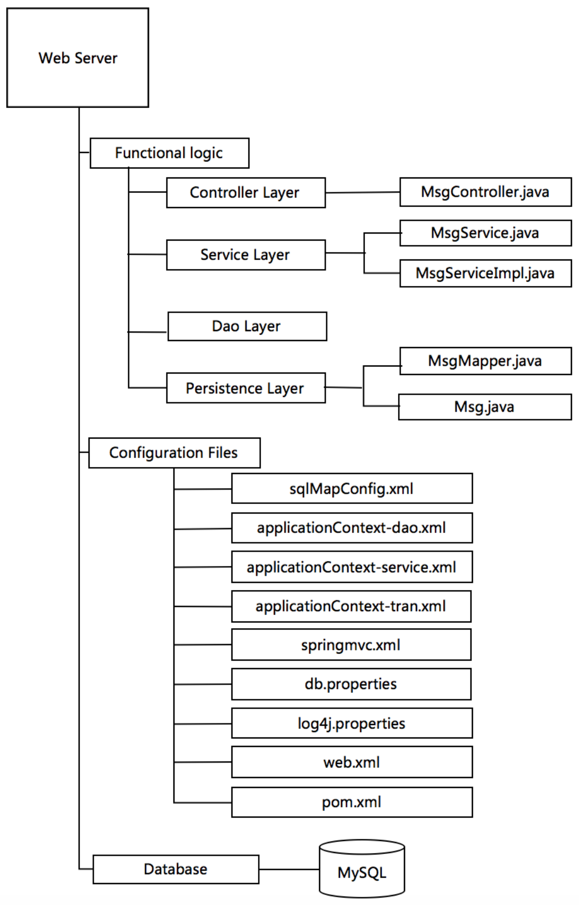

Fig. 5: Web Server side architecture

variables, which is generally expressed in the letter ρ. Because of different subjects, there are many ways to define correlation coefficients. In general, the formula for the correlation coefficient is [21]:

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

That is, the covariance of X and Y is divided by the standard deviation of X and the standard deviation of Y.

Therefore, the correlation coefficient can also be considered as a covariance. It is a special covariance that eliminates two dimensional variables and normalization. So, the correlation coefficient can reflect whether the two variables change in the same direction or in reverse direction. The correlation coefficient is positive if correlated in the same direction, or negative if in the reverse direction. It eliminates the influence of the variation of two variables, but simply reflects the similarity of two variables per unit change. The closer the correlation coefficient is to 1 or -1, the stronger the correlation is. The closer the correlation coefficient is to 0, the weaker the correlation is.

According to this experiment, we consider two kinds of correlation coefficient testing methods: spearman's rank correlation coefficient test and kendall's rank correlation coefficient test.

a) Spearman's rank correlation coefficient test: In statistics, Spearman's rank correlation coefficient, named after Charles Spearman and often denoted by the Greek letter ρ. It is a nonparametric index that measures the dependence of two variables. It uses monotone equation to evaluate the correlation of two statistical variables. If there is no duplicate value in the data, and when the two variables are completely monotonically correlated, the Spearman's correlation coefficient is +1 or 1. The Spearman's rank correlation coefficient is defined as the Pearson correlation coefficient between the ranked variables [23].

For a sample of size n, the n raw scores X_i,Y_i are converted to ranks rg_{Xi}, rg_{Yi}, and r_s is computed from:

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$$

where

- ρ denotes the usual Pearson correlation coefficient, but applied to the rank variables.
- cov(rg_X,rg_Y) is the covariance of the rank variables.
- σ_{rg_X} and σ_{rg_Y} are the standard deviations of the rank variables.

b) Kendall's rank correlation coefficient test: In statistics, the Kendall rank correlation coefficient, commonly referred to as Kendall's tau coefficient (after the Greek letter τ), is a statistic used to measure the ordinal association between two measured quantities. A tau test is a non-parametric hypothesis test for statistical dependence based on the tau coefficient.

Let (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n) be a set of observations of the joint random variables X and Y respectively, such that all the values of (x_i) and (y_i) are unique. Any pair of observations(x_i, y_i) and (x_j, y_j), where i $\neq$ j, are said to be concordant if the ranks for both elements (more precisely, the sort order by x and by y) agree: that is, if both $x_i > x_j$ and $y_i > y_j$; or if both $x_i < x_j$ and $y_i < y_j$.They are said to be discordant, if $x_i > x_j$ and $y_i < y_j$; or if $x_i < x_j$ and $y_i > y_j$. If $x_i = x_j$ or $y_i = y_j$, the pair is neither concordant nor discordant. The Kendall τ coefficient is defined as [24]:

$$\tau = \frac{(number\ of\ concordant\ pairs) - (number\ of\ discordant\ pairs)}{n(n-1)/2}$$

The denominator is the total number of pair combinations, so the coefficient must be in the range -1 $\leqslant \tau \leqslant$1.
where

- If the agreement between the two rankings is perfect (i.e., the two rankings are the same) the coefficient has value 1.
- If the disagreement between the two rankings is perfect (i.e., one ranking is the reverse of the other) the coefficient has value -1.
- If X and Y are independent,then we would expect the coefficient to be approximately zero.

- An explicit expression for Kendall's rank coefficient is

$$\tau = \frac{1}{n(n-1)} \sum_{i \neq j} sgn(x_i - x_j)\,\mathrm{sgn}(y_i - y_j)$$

3) Data analysis: Firstly the 385 sample data are extracted from the data set we capatured, each sample data contains the instantaneous network traffic of 6 processes, which are Chatspy, Browser, Google quick searchbox, Chrome, Wechat, Mobile QQ. The corresponding process IDs are 10113, 10004, 10023, 10033, 10098 and 10101 as shown in Table IX. Let's suppose that Pid 10113 is an unknown process to the detection program.

TABLE IX: Process Identify Number

PID	Process Name
10113	Chatspy
10004	Browser
10023	Google quick searchbox
10033	Chrome
10098	WeChat
10101	Mobile QQ

In our experiment, we treat other applications besides pid 10113 to be the known processes for users, such as chrome, QQ, WeChat and so on. The target is to test whether pid 10113 is a spy process which may monitor other applications' private records. Since we do not the exact target application, we need to take all the active running processes into consideration. In our experiment, we take the 5 most active processes for the data analysis phase. We take one process and try to test the correlation against all the other processes in a combinatorial way. For example, we pick the spy application pid 10113 for the one random variable. The other one should be the combination of all the rest processes. Since we have only 6 processes, the rest contains 5, and we need to divide into 5 cases which are $C_5^1, C_5^2, C_5^3, C_5^4, C_5^5$. Table X-XIV show the correlation test result in each of the case. Notice that for the demonstration purpose, we picked the spy application for one side of the correlation test, but in reality we may not sure which one it is. So we need to perform the experiment we have shown here 6 more times.

TABLE X: Correlation coefficient Test 1

unknown pid	pid	Spearmanr	Kendalltau
10113	10004	-0.0956	-0.0885
	10023	-0.0734	-0.0704
	10033	-0.0865	-0.0828
	10098	0.5785	0.5530
	10101	0.6775	0.6518

Since in our setting, the spy application is set to capture the message of both wechat and qq, the correlation will reach the strongest point when test against the combination of two, namely, the pid 10098 and 10101, which are the pid of wechat and qq accordingly. According to the last row of the Correlation coefficient Test 2 Table, we can see that the result of Spearmanr and Kendalltau reaches 0.9357 and 0.8794, which show very strong correlation, and also they are the highest values among all the tests here. This provides us with a very strong evidence that the process with PID 10113 has a strong correlation with the application of wechat and qq regarding the network traffic flow. The other system and application running processes are selected according to the activation ranking, since we make the assumption that sending back the data to the remote server will take network bandwidth. However, we need to point out that if the social network application is not active, then we may not be able to make an accurate detection. A second limitation of our current detection strategy is that we only consider that the real-

TABLE XI: Correlation coefficient Test 2

unknown pid	pid	Spearmanr	Kendalltau
10113	10004-10023	-0.1193	-0.1098
	10004-10033	-0.1131	-0.1031
	10004-10098	0.2273	0.2065
	10004-10101	0.2526	0.2305
	10023-10033	-0.1189	-0.1110
	10023-10098	0.3226	0.3013
	10023-10101	0.4052	0.3810
	10033-10098	0.3382	0.3164
	10033-10101	0.4319	0.4062
	10098-10101	0.9357	0.8794

[1] *-* is a combined process code that combines the instantaneous traffic flow value generated by 2 different processes.

TABLE XII: Correlation coefficient Test 3

unknown pid	pid	Spearmanr	Kendalltau
10113	10004-10023-10033	-0.1356	-0.1227
	10004-10023-10098	0.1829	0.1648
	10004-10023-10101	0.2080	0.1885
	10004-10033-10098	0.1728	0.1542
	10004-10033-10101	0.1825	0.16415
	10004-10098-10101	0.5290	0.47215
	10023-10033-10098	0.2054	0.1870
	10023-10033-10101	0.2697	0.2479
	10023-10098-10101	0.6752	0.6202
	10033-10098-10101	0.7133	0.6552

[1] *-*-* is a combined process code that combines the instantaneous traffic flow value generated by 3 different processes.

TABLE XIII: Correlation coefficient Test 4

unknown pid	pid	Spearmanr	Kendalltau
10113	10004-10023-10033-10098	0.1324	0.1172
	10004-10023-10033-10101	0.1430	0.1279
	10004-10023-10098-10101	0.4716	0.4180
	10004-10033-10098-10101	0.4373	0.3846
	10023-10033-10098-10101	0.5272	0.4732

[1] *-*-*-* is a combined process code that combines the instantaneous traffic flow value generated by 4 different processes.

TABLE XIV: Correlation coefficient Test 5

unknown pid	pid	Spearmanr	Kendalltau
10113	10004-10023-10033-10098-10101	0.3852	0.3365

[1] *-*-*-*-* is a combined process code that combines the instantaneous traffic flow value generated by 5 different processes.

time capturing and sending. The adversary could use different strategies such as caching the content and randomizing the sending time, which can help keep under the radar. We will target more sophisticated adversaries in our future works.

V. CONCLUSION AND DISCUSSION

Malware has become a serious threat on the android platform. Monitoring applications which are in the gray zone, could be used for various purposes, including the spy behavior which can leak the user's privacy. In this paper, we studied the technical principle of such monitoring application, especially we focus on the behavior of the monitoring softwares that spy the chatting softwares such as wechat, qq, whatsapp and line. We made our spy prototype by using the Accessibility API provided by Android. Then we investigate the strategies on how to efficiently detect the spy applications based on the statistical behavior. By taking advantage of the correlation test, we are able to accurately distinguish the chatting application spy processes among the others. As one of our future works, we will investigate how to defeat the spy application in a more sophisticated manner where the attacker can choose other sending strategies.

ACKNOWLEDGMENT

This work has been partly supported by the National Natural Science Foundation of China under Grant No. 61702212 and the research funds of CCNU from colleges basic research and operation of MOE under Grand No. CCNU16A05040.

REFERENCES

[1] The entire list of mobile spysoftware applications. https://www.top10spysoftware.com/apps, 2018. Accessed April 17, 2018.

[2] Nicolas Christin Timothy Vidas, Daniel Votipka. All your droid are belong to us: A survey of current android attacks. *WOOT*, 2011.

[3] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur, and Mauro Conti. Android security: A survey of issues, malware penetration, and defenses. *IEEE COMMUNICATION SURVEYS & TUTORIALS*, 17(2), SECOND QUARTER 2015.

[4] Bahman Rashidi and Carol Fung. A survey of android security threats and defenses. 6:335, 10 2015.

[5] Sascha Fahl, Marian Harbach, Thomas Muders, Bernd Freisleben, and Matthew Smith. Why eve and mallory love android: an analysis of android ssl (in)security. In *ACM Conference on Computer and Communications Security*, pages 5061, 2012.

[6] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. *Crowdroid: Behavior-Based Malware Detection System for Android.* 2011.

[7] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. Riskranker:scalable and accurate zero-day android malware detection. *In International Conference on Mobile Systems, Applications, and Services*, pages 281294, 2012.

[8] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. andromaly: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161190, 2012.

[9] Dong Jie Wu, Ching Hao Mao, Hahn Ming Lee, and Kuo Ping Wu. Droidmat: Android malware detection through manifest and api calls tracing. In *Information Security*, pages 6269, 2012.

[10] Yousra Aafer, Wenliang Du, and Heng Yin. *DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android.* Springer International Publishing, 2013.

[11] Suleiman Y. Yerima, Sakir Sezer, Gavin Mcwilliams, and Igor Muttik. A new android malware detection approach using bayesian classification. In *IEEE International Conference on Advanced Information NETWORKING and Applications*, pages 121 128, 2013.

[12] A. D Schmidt, R Bye, H. G Schmidt, and J Clausen. Static analysis of executables for collaborative malware detection on android. In *IEEE International Conference on Communications*, pages 15, 2009.

[13] Takamasa Isohara, Keisuke Takemori, and Ayumu Kubota. Kernel-based behavior analysis for android malware detection. In *Seventh International Conference on Computational Intelligence and Security*, pages 10111015, 2012.

[14] Justin Sahs and Latifur Khan. A machine learning approach to android malware detection. In *Intelligence and Security Informatics Conference*, pages 141147, 2012.

[15] Naser Peiravian and Xingquan Zhu. Machine learning for android malware detection using permission and api calls. In *IEEE International Conference on TOOLS with Artificial Intelligence*, pages 300305, 2014.

[16] Hugo Gascon, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. Structural detection of android malware using embedded call graphs. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 4554, 2013.

[17] Brandon Amos, Hamilton Turner, and Jules White. Applying machine learning classi- fiers to dynamic android malware detection at scale. In *Wireless Communications and Mobile Computing Conference*, pages 16661671, 2013.

[18] S. Y Yerima, S Sezer, and G Mcwilliams. Analysis of bayesian classification-based approaches for android malware detection. *Information Security Iet*, 8(1):2536, 2016.

[19] Karim O. Elish, Xiaokui Shu, Danfeng Yao, Barbara G. Ryder, and Xuxian Jiang. Profiling user-trigger dependence for android malware detection. *Computers & Security*, 49:255273, 2015.

[20] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, and Yang Liu. A multi-view context-aware approach to android malware detection and malicious code localization.*Empirical Software Engineering*, (6):153, 2017.

[21] Accessibility. http://www.android-doc.com/guide/topics/ui/accessibility/index.html, 2018. Accessed April 17, 2018.

[22] ZHOU Gairong, WANG Zhifu, WANG Yongxue, and WANG Jian. Probability and Statistics. Beijing: Higher Education Press, 2009.

[23] Jerome L Myers and Arnold D Well. *Research design and statistical analysis (2nd ed.).* L. Erlbaum Associates, 2010.

[24] R.B. Nelsen. Kendall tau metric. In *Encyclopedia of Mathematics*. Springer Science+Business Media B.V. / Kluwer Academic Publishers, 2001.

[25] WeChat. Tencent Technology (Shenzhen) Company Ltd. https://play.google.com/store/apps/details?id=com.tencent.mm, 2018. Accessed April 17, 2018.

[26] QQ. Tencent Technology (Shenzhen) Company Ltd. https://play.google.com/store/apps/details?id=com.tencent.mobileqq, 2018. Accessed April 17, 2018.

[27] WhatsApp Messenger. WhatsApp Inc. https://play.google.com/store/apps/details?id=com.whatsapp, 2018. Accessed April 17, 2018.

[28] LINE: Free Calls & Messages. LINE Corporation. https://play.google.com/store/apps/details?id=jp.naver.line.android&hl=en, 2018. Accessed April 17, 2018.

[29] Gupta, Vipul, et al. "Sizzle: A standards-based end-to-end security architecture for the embedded internet." Pervasive and Mobile Computing 1.4 (2005): 425-445.