Malicious Android malware Detection Scheme using Feature Based on Destination Servers and Level of SSL Server Certificate

A Thesis Submitted in Candidacy for

the Master of Engineering Degree

by

Hiroya Kato

Under the guidance of

Professor *Iwao Sasase*

School of Science for Open and Environmental Systems,

Graduate School of Science and Technology,

Keio University, Yokohama, Japan

March 2019

Abstract

Detecting Android malwares which send encrypted malicious payloads is imperative. As an Android malware detection scheme, I focus on the scheme leveraging the fact that there exists the difference of network traffic patterns between benign apps and malwares because useful network traffic pattern based features can be obtained without packet inspection. However, such features are easy to be disguised by attackers. Thus, the malwares whose traffic patterns are similar to benign ones can evade that scheme. In this paper, I propose a detection scheme using features based on destination servers and the level of SSL server certificate. Attackers tend to use an untrusted certificate to encrypt malicious payloads in many cases because passing rigorous examination is required to get a trusted certificate. Thus, I utilize SSL server certificate based features for detection since their servers tend to be untrusted. Furthermore, in order to obtain the more exact malicious features which are hard to be disguised, I introduce required permission based weight values because malwares inevitably require permissions regarding malicious actions. By computer simulation with real dataset, I show my scheme can improve detection performance. Furthermore, I clearly demonstrate that my scheme can deal with the malwares transferring encrypted malicious payloads and the shortcoming of the previous scheme.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis is carried out under the direction and guidance of Prof. Iwao Sasase of Keio University during 2017-2019. The author is deeply indebted to a number of individuals who helped me make this work possible. My sincere gratitude and deepest appreciation should be first given to Prof. Iwao Sasase of Keio University for his valuable suggestions, guidance and continuous encouragement throughout this work. My gratitude should be also given to Mr.Shuichiro Haruta for his encouragement and valuable comments on this work. I also would like to all the members of Sasase Laboratories, Department of Information and Computer Science, Keio University, for encouraging my work. Especially, I am deeply grateful to Mr.Shuichiro Haruta. He gave me continuous committed encouragements and valuable suggestions for two years.

Finally, I would like to express my gratitude to my family members who nourished me, encouraged me and gave their utmost support to fulfill my educational aims. That resulted in this work.

School of Open and Environmental Systems,
Guraduate School of Science and Technology,
Keio University

Hiroya Kato

Chapter 1

Introduction

Android is the most popular smartphone platform occupying 85% of market share in the world [1]. Unfortunately, smartphones running on Android system have become the main target of attackers due to its popularity. Most malwares send sensitive information such as contact lists, SMS messages, GPS and device information to external servers via network. The Android aaps released on Google Play which is the official store of apps are automatically evaluated because manual evaluation spends a lot of personnel expenses and more time. Since such evaluation cannot completely prevent malwares from spreading, users are under the risk of installing malwares. Thus, this circumstance results in the urgency of detecting Android malwares.

Existing solutions for detecting malwares are mainly classified into required resource based schemes [2–5] and network based schemes [6, 7]. Required resource based schemes are conducted by utilizing the features extracted from required resource such as permissions and application programming interface (API) calls. These schemes focus on the fact that malwares have to utilize required resource regarding malicious actions. Although required resource based schemes are useful, these schemes are subject to clever techniques such as dynamic code loading [8] and transformation attack [9] [10].

In order to cope with the limitations of required resource based schemes, some researchers propose network based schemes. Network besed schemes pay attention to the fact that there exists strong evidence of malwares in network traffic data since most malwares communicate with external network to conduct malicious actions. These schemes can detect the malwares which use network to conduct malicious actions by using features regarding network behavior of Android apps. Although, several network based schemes have been proposed, I pay attention to the scheme proposed by Garg et al. [7]. This is because that scheme can deal with encrypted traffic data. More and more malwares tend to interact with external networks by using encrypted traffic in order to hide malicious payloads. Thus, it is necessary to extract useful features from malwares without deep packet inspection. That scheme leverages the fact that there exists the difference of network traffic patterns between benign apps and malwares. Network traffic pattern based features extracted from traffic data of running apps are fed into machine learning classifier, and it detects malwares. Although that shcme is a promising approach due to applicability to encrypted traffic data, traffic pattern based features are easy to be disguised by attackers. Hence, the malwares whose traffic patterns are similar to that of benign apps can evade that scheme.

In order to detect such malwares by using the features that can deal with encrypted traffic data and are hard to be disguised, in this paper, I propose a detection scheme using features based on destination servers and the level of SSL server certificate. The main idea of our scheme is that malwares tend to communicate with untrusted destination servers in order to transfer encrypted malicious payloads such as sensitive information and malicious codes.

Untrusted servers can be identified on the basis of the level of SSL server certificate. In order to encrypt traffic data, attackers have to introduce SSL server certificates to their servers. Attackers tend to use an untrusted certificate to encrypt malicious payloads in many cases since passing rigorous examination process is required to obtain a trusted certificate. Thus, my scheme can detect malwares by using SSL server certificate based features since their servers tend to be untrusted. However, since benign apps also communicate with untrusted servers, the detection performance may be degraded. In order to eliminate such situation, I introduce required permission base weight values to SSL server certificate based features because malwares inevitably have to require the permissions regarding malicious actions. By doing this, my scheme can obtain the more exact features of malwares. The contributions of this paper are as follows:

The rest of this paper is constructed as follows. After this introduction, related works are introduced in Section 2. The previous scheme and the shortcoming of that scheme are explained in Section 3. Proposed scheme is described in Section 4. Various evaluation results are their interpretation are shown in Section 5. Finally, the conclusions of this paper are presented in Section 6.

Chapter 2

Related Work

In order to detect Android malwares, several schemes have been proposed . These schemes are divided into required resource based detection and network based detection. Required resource based detection is conducted by utilizing the features extracted from required resource such as permissions and application programming interface (API) calls. Meanwhile, network based detection is performed on the basis of the features regarding network behavior of Android apps. The representative schemes are explained in the following subsections.

2.1 Required resource based detection

Encl et al. [2] propose the scheme leveraging the combination of permissions required by an Android app. That scheme pays attention to the fact that malwares tend to require the permissions regarding sensitive actions such as accessing device information and personal information of users. Therefore, Android malwares can be detected by matching required permission. However, since benign apps may also require sensitive permissions, a large number of benign ones may be regarded as malwares by that scheme. This is why it is necessary to design the schemes which rely on other features.

Deshotels et al. [3] propose the scheme relying on API calls in detecting Android malwares. That scheme focuses on the fact that malwares abuse sensitive API call to conduct malicious operations. Malwares are classified in accordance with matching Android API calls against the signatures that identify malwares. However, that scheme cannot detect the malwares which conspire with another app by dynamic code loading such as installing another malicious app since none of sensitive API calls are used by the malwares.

In order to deal with the malwares colluding with another app, Xu et al. [4] propose the scheme that builds malware detection models based on Inter-Component Communication (ICC). The main idea of that scheme is that there exists the difference of ICC patterns between benign apps and malwares. While ICC is mainly utilized for internal communication in a benign app, malwares tend to communicate with other apps via ICC to conduct malicious operations. Thus, that scheme can detect the malwares which invalidate most existing required resources based detection schemes by leveraging the ICC based features. However, that scheme cannot detect the malwares which conduct simple actions such as sending sensitive information via network due to the absence of ICC related features.

Sun et al. [5] propose the scheme that leverages dynamic binder call graph. Binder is a kernel driver for inter-process communication between apps. That scheme focuses on the fact that the binder call graph of malwares is similar to that of existing malwares variants since most malwares are created by adding new malicious logic into existing malwares. Because all the communication in Android apps need to go through the binder, binder call graph can accurately represent malicious behavior of malwares. Hence, that scheme can detect malwares which conspire with another app and conduct simple actions by utilizing dynamic binder call graph for signatures matching algorithm. However, malwares can invalidate that scheme by transformation attack such as inserting dummy actions into malicious actions in order not to be similar to malicious signatures [11].

Although above-mentioned required resource based schemes can detect a part of malwares, they are subject to the clever techniques such as dynamic code loading [8] and transformation attack [9] [10]. Thus, exploring new malware detection schemes which rely on something except required resource is needed due to the limitations of required resource based detection schemes.

2.2 Network based detection

Network based detection schemes leverages the fact that there exists strong evidence of malwares in network traffic data since most malwares communicate with external network to conduct malicious actions.

Wang et al. [6] propose the scheme which utilizes the text semantic features of network traffic to develop an effective malware detection model. That scheme pays attention to the occurrence of words regarding sensitive information in HTTP header of malwares. Since malwares tend to use HTTP-POST/GET methods in order to send sensitive information such as "latitude", "longitude" and "imei" which is the unique identifier of the phone, semantic text features can be extracted from traffic flows of malwares. Therefore, that scheme can detect malwares which send sensitive information to external servers with high accuracy. However, the malwares that utilize HTTPS protocol to encrypt malicious payloads cannot be efficiently detected by that scheme.

In order to deal with encrypted traffic data, Garg et al. [7] propose the scheme focusing on the network traffic patterns of Android apps. That scheme leverages the fact that there exists the difference of network traffic patterns between benign apps and malwares. Network traffic pattern based features such as packets and data bytes transferred between origin and the destination are extracted from traffic data of running apps. That scheme can efficiently deal with the malwares which send encrypted data because useful traffic patterns based features are extracted without deep packet inspection. Although two above network based detection schemes are proposed, we pay attention to that scheme [7] because more and more malwares tend to interact with external networks by using encrypted traffic [6] [7]. I explain that scheme as the previous scheme in the next section.

Chapter 3

Previous Scheme

3.1 Overview of the previous scheme

In order to deal with encrypted data, the traffic pattern based features are extracted without deep packet inspection in the previous scheme [7]. The main idea of the previous scheme is that there exists the difference of network traffic patterns between benign apps and malwares. Malwares more frequently use HTTP-POST/GET methods for transmission of sensitive data at short intervals than benign apps. Furthermore, Malwares tend to repeatedly try to conduct malicious actions such as sending sensitive information and loading malicious codes because most of the time the servers of attackers are offline or shut down to escape detection. Thus, creating malwares to ensure that malicious actions succeed results in the unique traffic patterns of malwares. That scheme can detect malwares which communicate with other networks to conduct malicious actions by using the unique traffic patterns of malwares. The traffic patterns based features regarding four categories, namely DNS, HTTP, TCP and origin-destination based ones are extracted from traffic data of an app running on a real device. Finally, these features are fed into machine learning classifier such as Decision Tree and Random Forest for detection.

3.2 Shortcoming of the previous scheme

Although the features of the previous scheme can cope with encrypted traffic data, traffic pattern based features are subject to usage situations of apps. Hence, since attackers can disguise traffic patterns of malwares by creating malwares whose traffic patterns are similar to benign ones, the previous scheme cannot detect such malwares. In particular, that scheme is not applicable to repackaged malwares. Because a repackaged malware is created by injecting malicious components into an original app, the traffic pattern of a repackaged malware is similar to that of an original app. Thus, repackaged malwares can evade the previous scheme. In order to resolve the shortcoming, it is necessary to propose the features which can deal with encrypted traffic data and are hard to be disguised by attackers.

Chapter 4

Proposed Scheme

4.1 idea

My goal is to design the scheme utilizing the features that are applicable to encrypted traffic data and are hard to be disguised. In order to accomplish my goal, I propose a detection scheme using features based on destination servers and the level of SSL server certificate. The main idea of my scheme is that malwares tend to communicate with untrusted destination servers in order to transfer encrypted malicious payloads such as sensitive information and malicious codes. Untrusted destination servers can be judged by identifying the level of SSL server certificates. The level of SSL server certificate is divided into three types, namely Extended Validation (EV), Organization Validation (OV) and Domain Validation (DV) certificates. EV and OV certificates are highly trusted certificates, because passing the rigorous examination process such as certifying existence of organizations is required to acquire them. Meanwhile, since a DV certificate is an unreliable one due to lenient examination, the server which has a DV certificate is untrusted one. In order to encrypt traffic data, attackers have to acquire SSL server certificates. I argue that attackers tend to exploit a DV certificate to encrypt malicious payload. This is because it is hard for them to acquire EV and OV certificates due to rigorous examinations. Furthermore, EV and OV certificates are expensive in comparison to a DV certificate. Thus, attackers are not willing to introduce expensive certificates to their servers because the objective of them is earning money by selling sensitive information. For the above reasons, in many cases, since attackers have no choice but to introduce a DV certificate to servers of them, their servers tend to be untrusted. Hence, my scheme can detect malwares by focusing on the communication with untrusted servers of attackers.

In order to demonstrate my argument, I inspected the percentage of apps communicating with DV servers. As shown in Figure 4.1, there exists the difference of the percentage of communication with DV servers between benign apps and malwares.

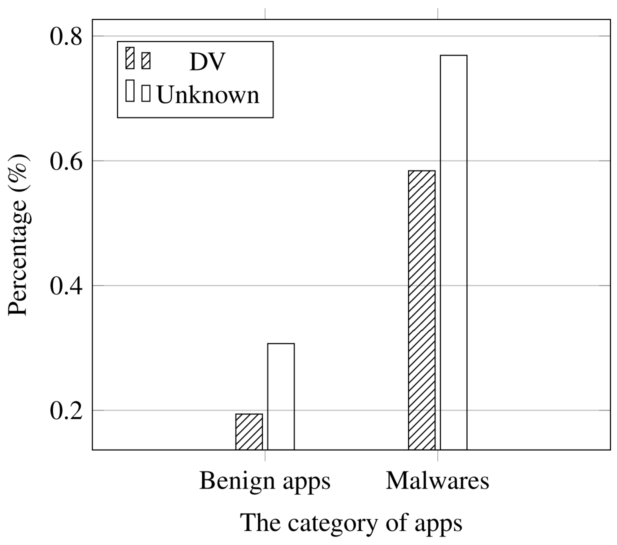

Figure 4.1: The result of inspection regarding SSL server certificate

Moreover, I discovered the servers whose the level of certificates is unknown because of refusing access to the servers or timeout (hereinafter, such servers are called "unknown servers"). As shown in Figure 4.1, in comparison to benign apps, malwares tend to communicate with the unknown servers. Thus, these results motivate us to extract SSL server certificate based features regarding DV and unknown servers. I can obtain the useful features which are hard to be disguised because attackers cannot easily introduce EV and OV certificates to the their servers. Furthermore, my features are applicable to encrypted data since they can be extracted without packet inspection. We mainly extract two types of features from each traffic data of an app. The first one is the total number of distinct IP address of untrusted servers, namely the DV and unknown servers because attackers tend to have several servers to evade detection. The second one is the feature based on the frequency of communication with DV and unknown servers. As already mentioned in the Section 3, malwares tend to repeatedly try to communicate with untrusted servers of attackers compared with benign apps. Hence, I utilize the frequency of communication with untrusted servers for malwares detection.

However, since benign apps also communicate with the untrussed servers, benign ones may be misjudged as malwares by using only SSL certificate based features. In order to cope with such situation, I focus on the fact that malwares inevitably have to require the permissions regarding malicious actions. For example, malwares require a permission, READ_PHONE_STATE for obtaining device information. By using such permission, malwares can send such data to untrusted servers of attackers. Meanwhile, an app which does not have such permissions cannot send device information to external servers. Hence, in order to exactly express malicious features, I introduce required permission based weight values to SSL certificate based features. By doing this, my scheme can reduce the misjudgement of the benign apps that communicate with untrusted servers. Finally, SSL certificate based features to which weight values are assigned are used for detection by machine learning classifier.

4.2 Algorithm

In this subsection, I explain the algorithm for extracting my features. The algorithm consists of four steps, namely network traffic collection, the level of SSL server certificate identification, SSL server certificate based feature generation and permission based weight assignment.

4.2.1 Network Traffic Collection

Network traffic data of an app is captured in this step. In order to capture the network traffic data, I execute an app on an Android emulator device provided by Android Studio [12]. I execute a single app at one time on an emulator device to ensure that the exact network traffic data generated by each app is captured. I then collect the network traffic packets of an app during the execution. The traffic data are saved as PCAP format files by using tcpdump command [13].

4.2.2 The Level of SSL Server Certificate identification

In this step, the level of SSL server certificate of each destination server is obtained to identify IP addresses of DV and unknown servers. In order to confirm the level of SSL server certificate, IP addresses are extracted from PCAP format file by using T-shark command [14]. I then obtain a SSL server certificate of the server which corresponds to each IP address by using openssl command [15]. In the EV and OV certificates, detailed geographic information of organizations is registered to certify the presence of organizations. Meanwhile, a DV certificate does not include such information since it is the certificate which authenticates only the authority for using domain. Thus, I can judge whether each destination server is a DV server by focusing on the information registered in a certificate. Furthermore, a server can be regarded as an unknown server when refusing access to the server or timeout errors occur. Accordingly, I can identify IP addresses of both DV and unknown servers from among all destination servers by doing above procedures.

4.2.3 SSL Server Certificate based Feature Generation

SSL server certificate based features are calculated on the basis of IP addresses of DV and unknown servers. I calculate the frequency of communication with DV and unknown servers and the total number of distinct IP addresses of DV and unknown destination servers. In terms of the frequency of communication, two types of the frequency are calculated by T-shark command. The first one is the frequency of communication for sending packets to an app from DV servers. The second one is the frequency of communication in the case where DV servers receive packets. Similarly, same procedures are conducted for IP addresses of unknown servers.

4.2.4 Permission based weight assignment

In this step, permission based weight value is assigned to SSL server certificate based features. First, I have to calculate permission based weight values on the basis of the occurrence of permissions in the malicious train dataset in advance.

I focus on only the dangerous permissions defined by the Android specification since these permissions are related to sensitive actions. Let $P_{\text{dangerous}} = \{p_i | 1 \leq i \leq 24\}$ denote the set of dangerous permissions. I create M_{p_i} which denotes the set of the malwares which request a dangerous permission p_i in M_{train}. Here, M_{train} is the set of all malwares in the training dataset. Then, I create the set of the weight value for each permission $W_{\text{permission}} = \{w_{p_i} | 1 \leq i \leq 24\}$. A weight value w_{p_i} for dangerous permission p_i is calculated as the following equation:

$$w_{p_i} = \frac{|M_{\text{p}_\text{i}}|}{|M_{\text{train}}|}. \tag{4.1}$$

I obtain P_{app} that indicates the sets of the dangerous permissions required by each app . P_{app} is represented as the following equation:

$$P_{\text{app}} = \begin{cases} \{p_k | 1 \leq k \leq 24\} & (|P_{\text{app}}| > 0), \\ \phi & (|P_{\text{app}}| = 0). \end{cases} \tag{4.2}$$

After that, w_{p_i} are assigned to SSL server certificate based feature f_{SSL} of each app on the basis of P_{app}. Accordingly, my features are calculated as the following equation:

$$f(f_{\text{SSL}}) = \begin{cases} f_{\text{SSL}} * (1 + \sum_{k=1}^{|P_{\text{final}}|} w_{p_k}) & (|P_{\text{final}}| > 0), \\ f_{\text{SSL}} & (|P_{\text{final}}| = 0). \end{cases} \tag{4.3}$$

Chapter 5

Evaluation

In order to demonstrate the effectiveness of my scheme, I evaluate Accuracy, True Positive Rate (TPR), and False Positive Rate (FPR) among the several schemes shown in Table 5.1 with real datasets. As shown in Table 5.1, the permission besed scheme utilizes only binary features regarding the presence of dangerous permissions. If an app has a dangerous permission, the feature regarding the permission is 1. Otherwise the feature is zero. Meanwhile, SSL certificate based scheme uses SSL server certificate based features without permission based weight. Accuracy, TPR, FPR are calculated as:

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{5.1}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{5.2}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{5.3}$$

where TP, TN, FP, and FN denote the number of True Positive (malwares are regarded as malwares), True Negative (benign apps are regarded as benign ones), False Positive (benign apps are regarded as malwares), and False Negative (malwares are regarded as benign apps), respectively.

Table 5.1: The schemes used for the evaluation

Schemes	The name of parameters used in our evaluation			
	SSL server certificate	weight value	Previous features	Dangerous permissions
My scheme	✓	✓		
Previous scheme			✓	
SSL certificate besed scheme	✓			
Permission besed scheme				✓

5.1 Simulation Parameters

Table 5.2 shows our simulation parameters. I use the apps from Androzoo [16] as the begin apps. Each of begin apps has been analysed by VirusTotal [20] which is an antivirus service with over 60 antivirus scanners. Thus, I regard the apps for which no antivirus scanners raise any alarm as begin ones. Meanwhile, malwares are collected from Androzoo and VirusShare [17] which is a repository of malware samples for security researchers. Since Androzoo provides the pairs of repackaged malwares and original apps, repackaged malwares are also collected from Androzoo dataset. In order to use apps as my datasets, network traffic data of them have to be obtained. Hence, I use only the apps which generate network traffic between 20 minutes packet monitoring as the my datasets. Accordingly, my dataset consists of 932 benign apps and 692 malwares. I utilize Random Forest classifier [18] for our evaluation because it is used in the previous scheme. Thus, I can fairly evaluate them. A series of experiments are conducted using ten-fold cross validation [19] to measure the performance of schemes in Table 5.1. Thus, the validity of the analysis can be certified in my simulation. I show the result of my evaluation for each scheme in Table 5.3.

Table 5.2: Simulation Parameters

The name of parameters	Value
Benign apps	Androzoo [16]
Malwares	Androzoo VirusShare [17]
The number of benign apps	932
The number of malwares	692
Classifier	Random Forest [18]
Validation	ten-fold cross validation [19]
Time of capturing traffic data	20 min.

Table 5.3: My evaluation result

Scheme	Accuracy(%)	TPR (%)	FPR (%)
My scheme	89.0	84.7	7.83
Previous Scheme	86.6	80.2	8.58
SSL certificate based scheme	86.6	83.2	10.1
Permission based features	86.6	76.2	5.69
My scheme & previous scheme	89.9	86.3	7.40

5.2 Comparison with the previous scheme

In order to show the effectiveness of my scheme, I compare my scheme with the previous scheme. As shown in Table 5.3, my scheme improve all of three criteria, namely accuracy, TPR and FPR. In particular, TPR is improved more than 4%. In order to demonstrate the reason, I inspect the malwares detected by my scheme but not by the previous scheme. After that, I discovered that the important previous features of these malwares are similar to that of benign apps. The importance of features can be calculated by using the Random Forest classifier. Thus, I regard the top five previous features to which high importance is assigned as the important previous features. Figure 5.1 shows the average values of the top 5 important previous features of the malwares detected only by my scheme.

As shown in Figure 5.1, these features of the malwares detected by my scheme are similar to the benign ones. My scheme can detect such malwares including repackaged malwares because strong evidence of malwares can be obtained from my features based on the relation between dangerous permissions and the level of SSL server certificates. Furthermore, I manually check SSL server certificate and traffic data of the malwares detected only by my scheme. After that, I discovered the malwares that communicate with an unknown server and send device information to the server. This result means that my scheme is applicable to encrypted data because the malwares which send sensitive information can be detected without packet inspection. Accordingly, I conclude that my scheme can deal with the shortcoming of the previous scheme and encrypted data. Furthermore, I evaluate the scheme combining my scheme and the previous scheme. As show in Table 5.3, the scheme achieves best detection performance in our evaluation. Hence, I conclude that

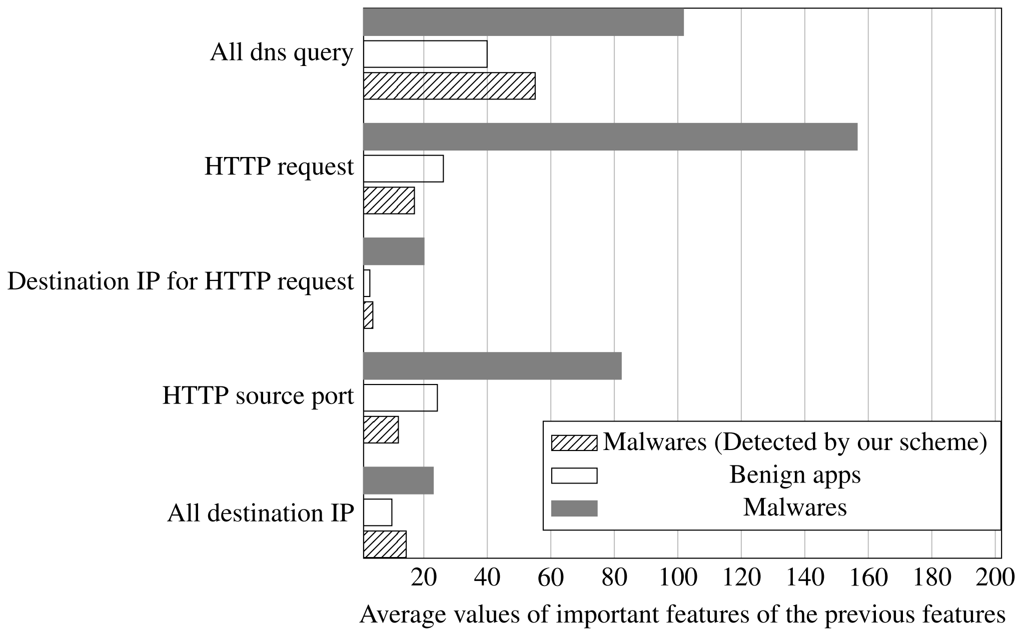

Figure 5.1: Average values of top 5 important previous features

my scheme and the previous scheme are complementary, and a hybrid scheme can more improve detection performance.

5.3 Evaluation of permission base weight values

In order to demonstrate whether permission based weight values are effective, I evaluate the detection performance of SSL certificate based scheme and the permission based scheme. As shown in Table 5.3, in comparison to my scheme, SSL certificate based scheme degrades detection performance regarding accuracy, TPR and FPR. Because the benign apps may communicate with DV and unknown servers, SSL certificate based scheme misjudges such benign apps as malwares. My scheme can correctly judge such benign apps by considering required permissions because they do not require dangerous permissions. Meanwhile, although the permission based scheme can correctly judge benign apps with low FPR, TPR is degraded in comparison to the other schemes. This is because that scheme is not applicable to repackaged malwares. After manually checking the dangerous permissions required by the repackaged malwares and original apps, I confirm that most repackaged malwares have the permissions required by original apps. My scheme can detect such repackaged malwares because my features indicate strong evidence of malwares by combining SSL server certificate based features and required dangerous permissions. Thus, I conclude that assigning permission based weight values to SSL server certificate based features is effective in improving detection performance.

5.4 False Positive Analysis

My scheme regards 73 benign apps as malwares in my simulation (i.e., false positives). After manually analyzing these false positives, I discovered that the app sending sensitive information such as IMEI which is the unique identifier of the phone to a unknown server while the app is labeled as benign one by VirusTotal. I conclude that the app is a malware since the app sends sensitive information to untrusted servers. I send the app to VirusTotal again in order to confirm the newest detection result of VirusTotal for the app because the detection results of VirusTotal have been updated continuously. Accordingly, 60 antivirus scanners in VirusTotal do not rise any alarm for the app. Therefore, the app is still regarded as benign one by VirusTotal. Furthermore, the previous scheme also labels the app as benign one. From these result, I conclude that my scheme can detect the malwares which evade 60 antivirus scanner and the previous scheme by using my features.

5.5 False Negative Analysis

My scheme misses 106 malwares, while 36 of them can be detected by the previous scheme. After manually checking the level of SSL server certificate and traffic data, I discovered that these malwares do not interact with untrusted servers. This is because I cannot bring out malicious actions of malwares. Some malwares do not conduct malicious actions on an emulator to escape detection. Thus, these inspection results reveal the limitation of my scheme.

Chapter 6

Conclusions

In this paper, I have proposed an Android malwares detection scheme using features based on destination servers and the level of SSL server certificate. I utilize SSL server certificate based features which are hard to be disguised. In order to obtain more exact malicious features, I introduce required permission based weight values. The detection performance of my scheme is better than the previous scheme. My evaluation results show that my scheme can deal with encrypted traffic data and the shortcoming of the previous scheme. Furthermore, after manually analysizing misjudged benign apps, I discover the malware which sends sensitive information to a untrusted server. The analysis results demonstrate that my scheme can detect malware that evade the detection of other antivirus scanners and the previous scheme.

References

[1] IDC, “Smartphone os market share, 2018 q3,” Available: https://www.idc.com/promo/smartphone-market-share/os.

[2] W. Enck, M. Ongtang, and P. McDaniel, “On lightweight mobile phone application certification,” in *Proceedings of the 16th ACM conference on Computer and communications security.* ACM, 2009, pp. 235–245.

[3] L. Deshotels, V. Notani, and A. Lakhotia, “Droidlegacy: Automated familial classification of android malware,” in *Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014.* ACM, 2014, p. 3.

[4] K. Xu, Y. Li, and R. H. Deng, “Iccdetector: Icc-based malware detection on android,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1252–1264, 2016.

[5] M. Sun, X. Li, J. C. Lui, R. T. Ma, and Z. Liang, “Monet: a user-oriented behavior-based malware variants detection system for android,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1103–1112, 2017.

[6] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, “Detecting android malware leveraging text semantics of network flows,” *IEEE*

Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1096–1109, 2018.

[7] S. Garg, S. K. Peddoju, and A. K. Sarje, "Network-based detection of android malicious apps," *International Journal of Information Security*, vol. 16, no. 4, pp. 385–400, 2017.

[8] A. Arora and S. K. Peddoju, "Ntpdroid: A hybrid android malware detector using network traffic and system permissions," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).* IEEE, 2018, pp. 808–813.

[9] M. Zheng, P. P. Lee, and J. C. Lui, "Adam: an automatic and extensible platform to stress test android anti-virus systems," in *International conference on detection of intrusions and malware, and vulnerability assessment.* Springer, 2012, pp. 82–101.

[10] V. Rastogi, Y. Chen, and X. Jiang, "Droidchameleon: evaluating android anti-malware against transformation attacks," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security.* ACM, 2013, pp. 329–334.

[11] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: a survey of issues, malware penetration, and defenses," *IEEE communications surveys & tutorials*, vol. 17, no. 2, pp. 998–1022, 2015.

[12] "Android studio," https://developer.android.com/studio/index.html.

[13] “The tcpdump & libpcap,” https://www.tcpdump.org.

[14] “Tshark-the wireshark network analyser,” http://www.wireshark.org.

[15] “Openssl cryptgraphy and ssl/tls toolkit,” https://www.openssl.org.

[16] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, “Androzoo: Collecting millions of android apps for the research community,” in *Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on*. IEEE, 2016, pp. 468–471.

[17] “virusshare.com,” https://virusshare.com.

[18] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19] R. Kohavi *et al.*, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[20] “Virustotal.com,” Available: https://virusshare.com.

Appendix A

List of the Related Papers by the Author

A.1 International Conference Papers

(1) **Hiroya Kato**, Shuichiro Haruta, and Iwao Sasase, "Malicious PDF Detection Scheme Using the Useful Feature Based on Non-Frequent Keywords in a File," in Proc. IEICE Information and Communication Technology Forum 2017 (ICTF2017), Poznan, Poland, July. 2017.

(2) Kyohei Osuge, **Hiroya Kato**, Shuichiro Haruta, and Iwao Sasase, "Feature Selection Scheme for Android ICC-related Features Based on the Gap of the Appearance Ratio," in Proc. IEICE Information and Communication Technology Forum 2018 (ICTF2018), Graz, Austria, July. 2018.

A.2 Technical Reports

(1) 加藤広野, 春田秀一郎, 笹瀬巌, ”ファジィ推論をキーワード及びその種類数に適用して 得られる特徴を用いた悪性 PDF 検知法,” 情報通信システムセキュリティ研究会 (ICSS), 長崎, 2017 年 3 月.

(2) 大菅恭平, 加藤広野, 春田秀一郎, 笹瀬巌, ”Android の ICC に関する特徴の出現率の差異 に基づいた特徴選択方式,” 通信方式研究会, 愛媛, 2018 年 11 月.