PAPER

Android Malware Detection Scheme Based on Level of SSL Server Certificate

Hiroya KATO[†a)], ***Member***, **Shuichiro HARUTA**[†b)], ***Student Member***, ***and*** **Iwao SASASE**[†c)], ***Fellow***

SUMMARY Detecting Android malwares is imperative. As a promising Android malware detection scheme, we focus on the scheme leveraging the differences of traffic patterns between benign apps and malwares. Those differences can be captured even if the packet is encrypted. However, since such features are just statistic based ones, they cannot identify whether each traffic is malicious. Thus, it is necessary to design the scheme which is applicable to encrypted traffic data and supports identification of malicious traffic. In this paper, we propose an Android malware detection scheme based on 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, we utilize SSL server certificate based features for detection since their certificates tend to be untrusted. Furthermore, in order to obtain the more exact features, we introduce required permission based weight values because malwares inevitably require permissions regarding malicious actions. By computer simulation with real dataset, we show our scheme achieves an accuracy of 92.7%. True positive rate and false positive rate are 5.6% higher and 3.2% lower than the previous scheme, respectively. Our scheme can cope with encrypted malicious payloads and 89 malwares which are not detected by the previous scheme.

key words: *Android malwares, SSL certificate, machine learning*

1. Introduction

Android is the most popular smartphone platform occupying 85% of market share in the world [1]. In that situation, unfortunately, smartphones running on Android system have become the main target of attackers due to its popularity [2]. Most malwares send sensitive information such as contact lists, SMS messages, GPS and device information to external servers via network. The Android apps 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 internal interaction based schemes [3]–[5] and unavoidable feature based schemes [6]–[8]. Internal interaction based schemes focus on internal behavior information such as application programming interface (API) calls and communication among Android system and apps.

Deshotels et al. [3] propose a scheme focusing on the fact that malwares abuse sensitive API call. Malwares are classified in accordance with malicious patterns predefined in advance. However, that scheme is difficult to detect the malwares which conspire with another app by installing another malware since none of sensitive API calls are used by the main malwares.

In order to deal with such collusion attack, Xu et al. [4] propose a scheme focusing on the difference of Inter-Component Communication (ICC) patterns between benign apps and malwares. ICC is inner communication among Android system and apps. 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. Although that scheme can detect the collusion attack, it is difficult for that scheme to 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 a scheme that leverages binder call graph (BCG) to detect malwares. Binder relays interprocess communication between apps. That scheme focuses on the fact that the BCG of malwares is similar to that of existing malwares variants since most malwares are created by adding new malicious logic into existing ones. Although that scheme is applicable to both collusion attacks and simple malicious ones, malwares can invalidate that scheme by inserting dummy actions into malicious actions in order not to be similar to BCG of existing malwares [9]. Although the schemes mentioned above are useful, their features can be manipulated by clever attackers' implementation techniques. Thus, since internal interaction based schemes can be evaded by such techniques, exploring new detection schemes is needed.

Meanwhile, unavoidable feature based schemes [6]–[8] focus on the fact that most malwares must exploit permissions regarding malicious operations and network to establish attacks. Because these features are independent of attackers' implementation techniques, useful features can be extracted from most malwares.

Li et al. [6] propose a scheme leveraging the fact that malwares tend to require the common permissions which enable to conduct high risk operations such as accessing device information and personal information of users. However, permissions are just features indicating potential maliciousness since it does not include dynamic evidence. Al-

Manuscript received May 1, 2019.
Manuscript revised September 25, 2019.
Manuscript publicized October 30, 2019.

[†]The authors are with Dept. of Information and Computer Science, Keio University, Yokohama-shi, 223–8522 Japan.
a) E-mail: kato@sasase.ics.keio.ac.jp
b) E-mail: haruta@sasase.ics.keio.ac.jp
c) E-mail: sasase@ics.keio.ac.jp

DOI: 10.1587/transinf.2019EDP7119

though permission are useful as the supplementary features, using only permissions is insufficient to accurately detect malwares. Thus, the features including dynamic evidence are needed.

Wang et al. [7] propose a scheme which pays attention to the occurrence of words regarding sensitive information in HTTP header of traffic data of malwares. Since malwares use HTTP-POST/GET methods for sending sensitive information, semantic text features can be extracted. The semantic text features are the words such as "latitude", "longitude" and "imei" which is the unique identifier of the phone. Although that scheme can extract useful features including dynamic evidence, the malwares that encrypt malicious payloads cannot be efficiently detected.

In order to deal with encrypted traffic data, Garg et al. [8] propose a scheme leveraging the difference of network traffic patterns between benign apps and malwares. Although various schemes are proposed, we pay attention to the scheme [8] as the previous scheme because unavoidable features including dynamic evidence can be extracted regardless of attackers' implementation techniques and encryption. However, the previous features are insufficient to precisely represent the features of malwares. Because the previous features are just statistic based ones, they cannot identify whether each traffic is malicious. Thus, it is necessary to design the scheme which enables to support identification of malicious traffic.

In order to support identification of malicious traffic, in this paper, we propose an Android malware detection scheme based on 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. As a result, the malicious evidences of traffic can be obtained. Untrusted servers can be identified on the basis of the level of SSL server certificate of destination servers. In order to encrypt traffic data, attackers must introduce SSL server certificates to their servers. Attackers tend to use an untrusted certificate in many cases since passing rigorous examination process is required to obtain a trusted certificate. However, since benign apps also communicate with untrusted servers, the detection performance may be degraded. In order to mitigate such situation, we introduce required permission based weight values to SSL server certificate based features because malwares must require the permissions regarding malicious actions. By doing this, our scheme can obtain the more exact features of malwares. The contributions of this paper are as follows:

1. Through an inspection regarding SSL server certificate of destination servers, we discovered that there exists the difference of the certificates introduced to destination servers between benign apps and malwares.
2. To the best of our knowledge, our scheme is first one which uses level of SSL server certificate for detection. Our scheme can be applicable to encrypted traffic data and support identification of malicious traffic.

The rest of this paper is constructed as follows. Attack model, the previous scheme and its shortcoming are introduced in Sect. 2. Proposed scheme is described in Sect. 3. Various evaluation results are shown in Sect. 4. Finally, the conclusions of this paper are presented in Sect. 5.

2. Attack Model and Previous Scheme

2.1 Attack Model

We assume attackers try to conduct malicious actions such as stealing sensitive information from user's devices and installing other malwares on user's device by exploiting Android malwares. In order to establish attacks, it is premised that malwares communicate with external attacker's servers via HTTP and HTTPS.

2.2 Previous Scheme

2.2.1 Overview of the Previous Scheme

The main idea of the previous scheme [8] is that there exists the difference of network traffic patterns between benign apps and malwares. For example, in terms of the communication of malwares, a large amount of communication occurs in a short period to finish the attack in a short time. In the literature [10] which carries out empirical analysis of lifetime of Command and Control (C&C) servers, an analytical result shows that survival probability of C&C servers becomes about 50% after 1,000 hours (42 days) of inspection. In many cases, attackers tend to use hosting servers as C&C servers. In the case where hosting servers are judged that they are abused as C&C ones, IP addresses of the C&C servers are blacklisted, and those servers become unavailable. Hence, the lifetime of C&C servers is limited and short compared with legitimate servers. Although the literature [10] is study about C&C servers, it is not irrelevant to the previous scheme since it is premised that malwares communicate with C&C servers in the previous scheme. Since the lifetime of attacker's servers is limited, malwares repeatedly try to conduct malicious actions regardless of user's operations to enhance the success probability of attacks. Thus, malwares more frequently transmit sensitive data at short intervals than benign apps. Table 1 shows the features used in the previous scheme. The previous scheme mainly utilizes four types of features, namely, DNS, HTTP, Origin-Destination (O-D) and TCP based ones. In particular, O-D based features are calculated from packets and data bytes transferred between user's device and destination servers. Since traffic pattern based features such as byte size and packet length are extracted only from traffic data of a running app, the useful ones including dynamic evidence can be extracted regardless of attackers' techniques and encryption. Finally, these features are fed into machine learning classifier such as Decision Tree and Random Forest for detection.

Table 1 The features used in the previous scheme.

Category	Features
DNS	1.All DNS query count, 2.Average DNS interval, 3.Distinct DNS query
HTTP	4.GET/POST request count, 5.GET/POST request interval, 6.Destination IP count for distinct GET/POST request, 7.Average packets length of GET/POST request, 8.Distinct source port used for GET/POST request
O-D	9.Average packet length, 10.Sent bytes size, 11.Received bytes size, 12.Average sent packet length, 13.Average received packet length, 14.Sent packet count, 15.Received packet count, 16.Destination IP count 17.IO bytes (sent bytes size / received bytes size),
TCP	18.TCP-RST packet count, 19.TCP-SYN packet count

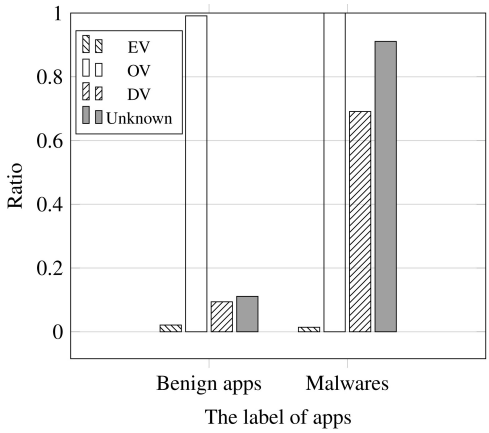

Fig. 1 The ratio of apps which communicate with each kind of server.

2.2.2 Shortcoming of the Previous Scheme

Although the previous features are useful ones which can be extracted regardless of clever attackers' techniques and encryption, they are insufficient to represent the features of malwares. Since the previous features are just statistic based ones, they cannot identify whether each traffic is malicious. Thus, it is necessary to identify malicious traffic for more exact evidence of malwares by focusing on the factor that attackers cannot eliminate. The requirements that we must satisfy are to be applicable to encryption and support identification of malicious traffic.

3. Proposed Scheme

In order to meet the requirements mentioned in Sect. 2.2.2, in this paper, we propose an Android malware detection scheme based on level of SSL server certificate. The main idea of our scheme is that malwares tend to communicate with untrusted servers in order to transfer encrypted malicious payloads. Untrusted servers can be identified by the level of SSL server certificates of destination servers. 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. For example, major enterprises introduce trusted certificates to their servers in order to indicate trust for themselves. Meanwhile, since a DV certificate is an untrusted one due to lenient examination, the server which has a DV certificate is not trusted one. In order to encrypt traffic data, attackers must acquire SSL server certificates. From this point of view, we argue that attackers tend to use a DV certificate to encrypt malicious payload. This is because it is hard for them to acquire trusted certificates due to rigorous examinations. Furthermore, the price of getting DV certificate is further lower in comparison to EV and OV certificates. Thus, attackers are willing to introduce a DV certificate to their servers because one of their objectives is earning money by selling sensitive information.

In order to demonstrate our argument, we first inspected SSL server certificate of destination servers by using our dataset. The data used for this inspection is based on the same data source described in Sect. 4.1 (Simulation). In Sect. 4.1, the data is described in detail. The method to inspect the certificate level is the same as the method of Sect. 3.1.2. In this inspection, we found the servers whose level of certificates is unknown because of refusing access to the servers or timeout. Hereinafter, we call such servers "Unknown servers". Figure 1 shows the ratio of apps which communicate with each kind of server. The x-axis of Fig. 1 indicates the label of apps, namely, benign label and malicious label. This ratio is separately calculated for benign apps and malwares. An app can be included in the ratio regarding communication with each kind of server. For example, suppose that a malware communicates with OV and DV servers, the malware is included in the numerator of the ratio regarding both OV and DV in duplicate. The result of Fig. 1 means that malwares tend to communicate with DV and Unknown servers in comparison to benign apps. Meanwhile, in terms of EV and OV, there is no difference between benign apps and malwares. Although there exists the case where malwares communicate with all kind of servers, namely, EV, OV, DV and Unknown servers, we concluded that our scheme can support identification of malicious traffic by focusing on the communication with DV and Unknown servers (hereinafter, such communication are called "untrusted communication") from the result of Fig. 1. Thus, these results motivate us to use SSL server certificate based features (hereinafter, such features are called "SSL features").

Furthermore, our scheme supports malwares that communicate with the attacker's server only by HTTP. A server that only supports HTTP is included as an Unknown server since the SSL server certificate cannot be obtained from such server. From the viewpoint of a server's reliability, since a server that does not have certificate is not trusted, the communication with such server is handled as untrusted communication in our scheme. In order to deal with the malwares which communicate with such servers, our scheme

Table 2 SRDP and RRDP.

Category	Permission name
SRDP	1.READ_PHONE_STATE, 2.READ_CALENDAR, 3.READ_CALENDAR, 4.READ_CONTACTS, 5.GET_ACCOUNTS, 6.ACCESS_FINE_LOCATION, 7.ACCESS_COARSE_LOCATION, 8.CAMERA, 9.READ_PHONE_STATE, 10.READ_CALL_LOG, 11.ADD_VOICEMAIL, 12.RECORD_AUDIO, 13.BODY_SENSORS, 14.SEND_SMS, 15.READ_SMS, 16.READ_EXTERNAL_STORAGE
RRDP	1.WRITE_CALENDAR, 2.WRITE_CONTACTS, 3.WRITE_EXTERNAL_STORAGE, 4.WRITE_CALL_LOG, 5.RECEIVE_SMS, 6.RECEIVE_MMS, 7.RECEIVE_WAP_PUSH

Table 3 SRF and RRF used in our scheme.

Category		Protocol	Features
SRF	USRF	HTTP	1.GET/POST request count, 2.GET/POST interval, 3.GET/POST destination count, 4.HTTP source port count
		All	5.Sent bytes size, 6.Sent packet count, 7.Average sent packet length, 8.Destination IP count
	TSRF	All	9.Ratio of sending to trusted servers
		HTTPS	10.Ratio of HTTPS sending to trusted servers
RRF	URRF	HTTP	11.Average received packet length, 12.Received bytes size, 13.Received packet count
		TCP	14.Number of TCP packet with RST bit
	TRRF	All	15.Ratio of receiving from trusted servers
		HTTPS	16.Ratio of HTTPS receiving from trusted servers

adopts HTTP related features. The proposed HTTP related features are calculated from traffic data regarding the untrusted communication with Unknown servers. In addition to that, our scheme is applicable to encrypted data since IP address is not encrypted.

However, since benign apps also communicate with DV and Unknown servers, benign ones might be misjudged as malwares. In order to mitigate such situation, we introduce required permission based weight values to SSL features. This is because malwares inevitably require the dangerous permissions (DP) [11]. DP are predefined as the ones regarding access to sensitive information in Android reference. By considering required DP, our scheme can extract more exact features and reduce the misjudgement of benign apps which do not require DP. In order to properly assign required DP based weight to SSL features, we divide DP into two types, namely sending related DP (SRDP) and receiving related ones (RRDP). For example, READ_PHONE_STATE is SRDP since a malware which sends device information to attackers' servers requires it for accessing device information. Table 2 shows SRDP and RRDP.

Moreover, our features are also divided into sending related features (SRF) and receiving related ones (RRF) in accordance with relation with SRDP and RRDP. Table 3 shows SRF and RRF. SRF consist of untrusted SRF (USRF) and trusted SRF (TSRF). Furthermore, RRF are also divided into untrusted RRF (URRF) and trusted RRF (TRRF). In our scheme, there are new designed features and the features based on the ones of the previous scheme. The features of TSRF and TRRF listed in Table 3 are new designed features. Meanwhile, the features of USRF and URRF listed in Table 3 are based on the ones in the previous scheme. They are selected from the features of the previous scheme in accordance with the manner whether the reliability of a destination server can be associated with the features. For example, HTTP based features are used in the previous scheme. This type of feature is adopted in our scheme because we can associate Unknown servers with the features by confirming that any SSL server certificate is not introduced to them. Although the previous scheme also utilizes DNS based features, they are not adopted in our scheme. This is because DNS is not related to the reliability of destination servers. Furthermore, since required DP based weight is applied to the proposed features, the values of them are completely different from those of the features in the previous scheme. USRF are based on untrusted communication. SRDP based weight is assigned to only USRF. This is because only the correlation between DP and untrusted communication can be malicious evidence. Suppose a malware sends device information to untrusted attackers' servers and requires READ_PHONE_STATE and WRITE_EXTERNAL_STORAGE. In this case, USRF and READ_PHONE_STATE in SRDP can be useful for detecting this malware. Meanwhile, WRITE_EXTERNAL_STORAGE in RRDP is not related to the malicious sending. Hence, only READ_PHONE_STATE based weight should be assigned to USRF. Similarly, the RRDP based weight is applied to URRF. By doing this, our scheme can properly assign required DP based weight to SSL features.

TSRF are based on trusted communication with EV and OV servers. TSRF consists of "9.Ratio of sending to trusted servers" and "10.Ratio of HTTPS sending to trusted servers". "9.Ratio of sending to trusted servers" is the ratio of sending to trusted servers to all sending. This feature is calculated on the basis of packet sending by not only HTTP but also all communication protocols including TCP and IP and so on. Since apps inevitably communicate with servers by other protocols such as TCP before HTTP or HTTPS communication, we consider all protocols. A trusted server can also perform HTTP communication in principle. In the case where HTTP communication is supported in trusted server setting, and it receives HTTP requests, it can deal with HTTP communication. However, a trusted server seldom performs HTTP communication because it does not have to intentionally support HTTP communication. On the other hand, "10.Ratio of HTTPS sending to trusted servers" is the ratio of HTTPS sending to trusted servers to all HTTPS sending. We also calculated features of TRRF by same strategy used for TSRF. Finally, our features in Table 3 are used for detection by machine

learning classifier.

3.1 Algorithm

In this section, the algorithm for extracting our features is explained. The algorithm consists of four procedures, namely 1) network traffic collection, 2) the level of SSL server certificate identification, 3) SSL features calculation, and 4) weighting feature. The procedures are repeatedly conducted for every app in $APP_{\text{all}} = \{a_i | 1 \leq i \leq n_{\text{all}}\}$, where n_{all} is the number of all apps. Let $M_{\text{train}}, B_{\text{train}}, M_{\text{test}}$, and B_{test} denote the training dataset of malwares, that of benign apps, test dataset of malwares and that of benign apps, respectively. We can also represent APP_{all} as $APP_{\text{all}} = M_{\text{train}} \cup B_{\text{train}} \cup M_{\text{test}} \cup B_{\text{test}}$.

3.1.1 Network Traffic Collection

Let traffic_{a_i} denote traffic data of a_i captured during the execution time. For the following procedures, traffic_{a_i} is saved in this step.

3.1.2 The Level of SSL Server Certificate Identification

This step extracts IP_{a_i} which is the set of IP addresses from traffic_{a_i}. In order to obtain SSL server certificate of extracted IP address, we use "openssl" command. Specifically, we accessed each IP address in IP_{a_i} by giving command "openssl s_client -connect [IP address]:443" and obtained SSL certificate of the server that corresponds to each IP address in IP_{a_i}. After obtaining SSL server certificate, the level of SSL server certificate can be identified as follows: (A) in the EV and OV certificates, detailed geographic information of organizations is included, (B) a DV certificate does not include such information, and (C) an Unknown server is identified in the case where refusing access to the server or timeout errors occur. Accordingly, the set of IP addresses of EV and OV servers, and the set of IP addresses of DV and Unknown servers are created. We denote them as $IP^{\text{T}}_{a_i}$ and $IP^{\text{U}}_{a_i}$, where T and U denote the labels of trusted and that of untrusted.

3.1.3 SSL Feature Calculation

Here, we only describe the features related to sending, namely USRF and TSRF. URRF and TRRF can be created by the same procedure. From traffic_{a_i} with $IP^{\text{T}}_{a_i}$ and $IP^{\text{U}}_{a_i}$, the features are counted or calculated. We denote the set of USRF and that of TSRF as $USRF_{a_i} = \{f_{j,a_i} | 1 \leq j \leq n_{\text{USRF}}\}$ and $TSRF_{a_i} = \{f'_{k,a_i} | 1 \leq k \leq n_{\text{TSRF}}\}$, where n_{USRF} and n_{TSRF} are the number of types of USRF and that of TSRF, respectively. Furthermore, when the above procedures are done in all apps, from M_{train}, $USRF_{\text{median}} = \{f^{\text{median}}_j | 1 \leq j \leq n_{\text{USRF}}\}$ which denotes the set of median of USRF is created.

3.1.4 Weighting Feature

Here, we only describe the weighting procedure for USRF. Weighting procedure for URRF can be represented by replacing "SRDP" and "USRF" appearing in following parameters to "RRDP" and "URRF", respectively. Let the set of SRDP denote $SRDP = \{p_l | 1 \leq l \leq n^{\text{SRDP}}\}$, where n^{SRDP} denotes the number of types of SRDP. $SRDP_{a_i}$ which denotes the set of existence of SRDP required by a_i is created as

$$SRDP_{a_i} = \{p_{l,a_i} | 1 \leq l \leq n^{\text{SRDP}}\}, \tag{1}$$

where p_{l,a_i} is 1 in the case where a_i requires p_l, otherwise it is 0. The required SRDP based weight $W^{\text{SRDP}}_{a_i}$ for a_i is calculated as

$$W^{\text{SRDP}}_{a_i} = n^{\text{SRDP}}_{a_i} + w^{\text{SRDP}}_{a_i}, \tag{2}$$

where $n^{\text{SRDP}}_{a_i}$ denotes the number of p_{l,a_i} which is 1. The larger $n^{\text{SRDP}}_{a_i}$ is, the more a_i has the capability to perform malicious action. $w^{\text{SRDP}}_{a_i}$ is calculated as

$$w^{\text{SRDP}}_{a_i} = \sum_{h=1}^{n^{\text{SRDP}}_{a_i}} P_{\text{malicious}}(p_{h,a_i}) - \sum_{h=1}^{n^{\text{SRDP}}_{a_i}} P_{\text{benign}}(p_{h,a_i}), \tag{3}$$

where $P_{\text{malicious}}(p_{h,a_i})$ is the function that returns probability of malwares requiring $p_h \in SRDP$. $P_{\text{malicious}}(p_{h,a_i})$ is calculated as

$$P_{\text{malicious}}(p_{h,a_i}) = \begin{cases} \frac{n^{\text{malicious}}_{p_h}}{n^{\text{malicious}}_{\text{train}}} & (p_{h,a_i} = 1), \\ 0 & (p_{h,a_i} = 0), \end{cases} \tag{4}$$

where $n^{\text{malicious}}_{p_h}$ and $n^{\text{malicious}}_{\text{train}}$ denotes the number of malwares requiring p_h and that of malwares in M_{train}, respectively. Similarly, the function $P_{\text{benign}}(p_{h,a_i})$ which returns the probability regarding benign apps can also be represented by replacing "malicious" in Eq. (4) to "benign".

For $f_{j,a_i} \in USRF_{a_i}$, a weighted feature $f^{\text{weighted}}_{j,a_i}$ is calculated as

$$f^{\text{weighted}}_{j,a_i} = \frac{1}{1 + \exp\{-\alpha_{a_i}(f_{j,a_i} - \frac{f^{\text{median}}_j}{2})\}}. \tag{5}$$

We introduce sigmoid function for calculating $f^{\text{weighted}}_{j,a_i}$ since the probability that a_i is a malware can be represented. $f^{\text{weighted}}_{j,a_i}$ takes the value between 0 and 1. The closer $f^{\text{weighted}}_{j,a_i}$ gets to 1, the more malicious a_i is. α_{a_i} is gradient of sigmoid function and changes in accordance with the proportion of untrusted communication. α_{a_i} is computed as

$$\alpha_{a_i} = (1 + \frac{s^{\text{USRF}}_{a_i}}{s^{\text{USRF}}_{\text{total}}}) \times (1 + \frac{h^{\text{USRF}}_{a_i}}{h^{\text{USRF}}_{\text{total}}}) \times W^{\text{SRDP}}_{a_i}, \tag{6}$$

where $s^{\text{USRF}}_{a_i}$ and $s^{\text{USRF}}_{\text{total}}$ are the frequency of sending to untrusted servers and that of total sending by a_i, respectively. Similarly, $h^{\text{USRF}}_{a_i}$ and $h^{\text{USRF}}_{a_i}$ denote same meaning regarding HTTPS communication, respectively. Finally, $USRF^{weighted}_{a_i} = \{f^{\text{weighted}}_{j,a_i} | 1 \leq j \leq n_{\text{USRF}}\}$ which denotes the set of weighted USRF is obtained.

4. Evaluation

In order to demonstrate the effectiveness of our scheme, we evaluate Accuracy (ACC), True Positive Rate (TPR), and False Positive Rate (FPR) with real dataset. ACC, TPR, FPR are calculated as

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{7}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{8}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{9}$$

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 4 shows the schemes used in the evaluation. We mainly compare two our schemes with the previous scheme and simple DP based scheme (SDPBS). Prop.1 is our scheme using SSL features to which required DP based weight is assigned. Prop.2 is also our scheme without required DP based weight. Prev. is the previous scheme, and SDPBS utilizes only binary features regarding the presence of DP. If an app has a permission in DP, the feature is 1. Otherwise the feature is zero.

4.1 Simulation

Table 5 shows our dataset. In order to obtain recent trend regarding the difference of characteristic between benign apps and malwares, we used latest dataset. Benign apps and repackaged malwares were obtained from Androzoo dataset. Each of benign apps has been analysed by VirusTotal [20] which is an antivirus service with over 60 antivirus scanners. Thus, we regard the apps for which no antivirus scanners raise any alarm as benign ones. We got latest benign apps and repackaged malwares as of February 2018. Furthermore, the other malwares were obtained from VirusShare dataset [13] which is a repository of malware samples for security researchers in February 2018. In order to use the obtained apps as our dataset, we must obtain traffic data by running them. Table 6 shows environment and tools used for 20 minutes traffic data collection. Each app is run on an Android emulator [14]. We use the Android tool called Monkey [15] in order to trigger malicious actions. The Monkey tool can randomly send events to the Android device when the app is running. Tcpdump [16] command is used to collect traffic data. However, running all of the obtained apps is not realistic since there exist a huge number of apps including benign apps and malwares. Thus, we randomly selected 1,000 malwares from the obtained ones because about 1,000 malwares are utilized for the evaluation in the paper of the previous scheme. Furthermore, in order to keep the proportion of benign apps and malwares in our dataset 1:1, 1,000 benign apps were also randomly selected. Selected apps were run on an Android emulator to obtain network traffic data. However, there exist the apps which do not generate network traffic data due to a failure of installing or clash during running them. Such apps were excluded from our dataset. We adopted only the apps which successfully run on an emulator and generate network traffic during 20 minutes packet monitoring as our dataset. Hence, because our dataset construction spent a lot of time, constructing large dataset was very difficult. Finally, our dataset consists of 801 benign apps and 884 malwares.

Table 7 shows tools used for generating features. T-shark [17] command is used for generating the proposed features and the features of the previous scheme. OpenSSL [18] command is used for obtaining SSL server certificate.

Table 8 shows a classifier and a validation method used in our simulation. We utilize Random Forest classifier [19] for our evaluation because it is used in the previous scheme. Our evaluation is conducted using ten-fold cross validation to guarantee the validity of the analysis in our simulation.

4.2 Comparison with the Previous Scheme

First, in order to show the effectiveness of our scheme, we compare Prop.1 with Prev. and the scheme combining Prev.

Table 4 The schemes used for the evaluation.

Schemes	The name of parameters used in our evaluation			
	SSL features	Weight assignment	Previous features	DP
Prop.1	✓	✓		✓
Prop.2	✓			
Prev.			✓	
SDPBS				✓

Table 5 Our dataset.

	Malwares	Benign apps
Source	Androzoo [12] VirusShare [13]	Androzoo
The number of apps	884	801

Table 6 Environment and tools used for 20 minutes traffic data collection.

Purpose	Tool
Environment for running apps	Android emulator [14]
Malicious action trigger	Monkey Tool [15]
Traffic data collection	Tcpdump [16]

Table 7 Tools used for generating features.

Purpose	Tool
Extracting features	T-shark [17]
Obtaining SSL server certificate	OpenSSL [18]

Table 8 Classifier and validation method used in our simulation.

	Name
Classifier	Random Forest [19]
Validation	ten-fold cross validation

Table 9 Evaluation result.

Scheme	ACC (%)	TPR (%)	FPR (%)
Prop.1	92.7	93.8	8.48
Prop.2	91.9	96.1	12.6
Prev.	88.2	88.2	11.7
SDPBS	84.1	79.1	10.3
Prev. & SDPBS	91.1	90.1	7.74
Prop.1 & Prev. & SDPBS	**93.9**	**94.3**	**6.49**

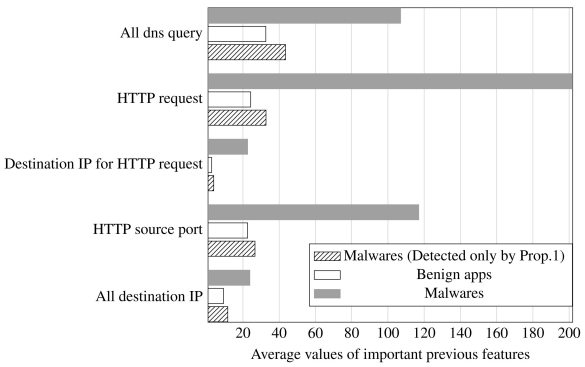

Fig. 2 The important previous features and their average values in malwares detected only by Prop.1, all benign apps, and all malwares.

and SDPBS. Table 9 shows the result of our evaluation for each scheme. As shown in Table 9, Prop.1 achieves the best ACC and TPR among three schemes. In particular, in comparison to Prev., TPR is improved up to 5.6%, and Prop.1 detects the 89 malwares not detected by Prev. Although FPR of Prop.1 is 0.74% higher than the scheme combining Prev. and SDPBS, it is minor difference. In order to more study, we inspect the malwares detected by Prop.1 and not detected by Prev. We calculate the importance of the previous features by using the Random Forest classifier which can express the importance of features as numerical value. We regard the top five previous features to which high importance is assigned as the important previous features. Figure 2 shows the important previous features and their average values in malwares detected only by Prop.1, all benign apps, and all malwares. As shown in Fig. 2, these features of the malwares detected only by Prop.1 are similar to the benign apps. Thus, it is natural that Prev. cannot detect them. Meanwhile, Prop.1 can detect such malwares because malicious traffic can be identified by focusing on SSL server certificate. We manually check SSL server certificate and traffic data of them. After that, we found the malwares that communicate with an Unknown server and send device information to the server. This result means that our scheme is applicable to the malwares which encrypt malicious payloads because the malwares can be detected without packet inspection. Accordingly, we conclude that Prop.1 can deal with the shortcoming of the previous scheme and encrypted data.

Furthermore, we evaluate the scheme combining Prop.1, Prev. and SDPBS. As show in Table 9, the scheme "Prop.1 & Prev. & SDPBS" achieves best performance. Hence, we conclude that a hybrid scheme can more improve detection performance.

4.3 Capability for Detecting the Malwares with Extremely Low Communication Frequency

Our scheme can detect the malwares with extremely low communication frequency. In such a case, the communication with attacker's untrusted server accounts for a large part of the communication since it must establish malicious actions with a few attempts. Due to a low amount of communication traffic, the features regarding the communication frequency might not indicate malicious evidence. However, the ratio of trusted communication to all communication tends to be low due to little trusted communication. On the other hand, such ratio of benign apps tends to be high. This difference enables to detect malwares with extremely low communication frequency. In fact, we detected the malware with low communication frequency. This malware only sent 3 HTTP POST requests and a HTTP GET request although the average of HTTP requests by malwares is 235 requests. Although the HTTP communication frequency of this malware is low, it is detected by our scheme. This is because the ratio of trusted communication to all communication is about 0.6, which is small ratio compared with benign apps of 0.9. Thus, we conclude that our scheme can detect the malware with extremely low communication frequency by using the ratio based features (These features are listed in TSRF and TRRF of Table 3 in Sect. 3). As defined in our attack model, in the case where malwares communicate with attacker's servers by HTTP and HTTPS, our scheme can deal with such malwares.

4.4 Evaluation of DP Based Weight Values

In order to demonstrate whether DP based weight value is effective, we compare Prop.1, Prop.2 and SDPBS. As shown in Table 9, in comparison to Prop.1, Prop.2 degrades ACC and FPR. In particular, FPR of Prop.2 is roughly 4% higher than Prop.1. This is because Prop.2 misjudges the benign apps which communicate with DV and Unknown servers as malwares. Meanwhile, Prop.1 can correctly judge such benign apps by considering required DP because they do not require DP. Furthermore, the performance of SDPBS is degraded in all metrics in comparison to Prop.1. This is because many benign apps have DP. Hence, this result means that using only required DP based features is insufficient to detect malwares since the malwares which have DP required by benign apps might be misjudged. In particular, it is difficult for SDPBS to detect repackaged malwares. Because a repackaged malware is created by injecting malicious codes into a benign app, its required DP is identical to the DP required by the benign app. After manually checking the DP required by the repackaged malwares and original apps, we confirm that most repackaged malwares have same DP required by original apps. Accordingly, SDPBS cannot correctly judge benign apps and repackaged malwares. Prop.1

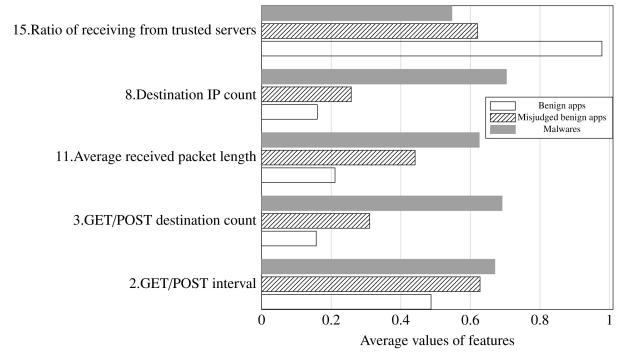

Fig. 3 The important proposed features and their average values in all benign apps, misjudged benign apps, and all malwares.

can also detect such malwares since our features indicate strong evidence of malwares by combining SSL features and required DP. Thus, we conclude that DP based weight values is effective in improving detection performance.

4.5 False Positive Analysis

Prop.1 regards 68 benign apps as malwares (i.e., false positives). In order to reveal the reason, we inspect our features which are important for detection. In order to identify the useful features, we utilize Random Forest classifier. Accordingly, from the proposed features in Table 3, we regard the top five proposed features to which high importance is assigned as important proposed features. In our simulation, the proposed features whose numbers are 2, 3, 8, 11 and 15 in Table 3 are regarded as important proposed features. Figure 3 shows the important proposed features and their average values in all benign apps, misjudged benign apps, and all malwares. As shown in Fig. 3, except for "15.Ratio of receiving from trusted servers", the average values of misjudged benign apps are larger than that of benign ones. Moreover, the values regarding "2.GET/POST interval" and "15.Ratio of receiving from trusted servers" are similar to the ones of malwares. This result means that misjudged benign apps frequently communicate with untrusted servers and require DP. Hence, we conclude that Prop.1 regards such apps as malwares since they communicate with untrusted servers.

In the case where benign apps which require DP communicate with untrusted servers, it is difficult for Prop.1 to correctly judge them as benign apps. However, there exist complementary improvement schemes which are promising for preventing FP in some cases. In the case where a benign app communicates with untrusted servers by HTTP, the scheme which conducts packet inspection [7] is useful. Since that scheme can reveal whether sensitive information is sent to external servers, it is promising for preventing FP. If a benign app does not send sensitive information, it can be correctly judged as benign one. On the other hand, in the case where a benign app communicates with untrusted servers by HTTPS, the schemes which focusing on Android architecture such as ICC [4] and API call [3] are useful. In order to obtain robust features which are independent of attacker's implementation techniques, Prop.1 focuses only on the understandable characteristic such as network traffic and permissions. Thus, Android architecture based features is not considered in Prop.1. In particular, the ICC based scheme and API call based one are promising for complementing Prop.1 because they are useful to judge whether an app has malicious functionality. For example, if a benign app does not have the functionality for sending sensitive information, it cannot conduct malicious actions such as privacy leakage. In this case, even if a benign app communicates with untrusted servers, it should not be judged as malware. Hence, since Android architecture based schemes can help reduce FP, combining Prop.1 and them is promising as countermeasures.

Furthermore, we manually analyze false positives to demonstrate whether there exist the malwares among them. After that, we found 2 benign apps sending sensitive information such as MAC address and phone number to DV and Unknown servers. The previous scheme regards these apps as benign ones. We resent them to VirusTotal in order to get latest detection results of these benign apps. The detection results of VirusTotal can be changed since it is updated and corrected over time. The 2 benign apps had been extracted from Androzoo in February 2018. After resending the 2 apps, they are received alarms from at least one of antivirus scanners in April 2019. Thus, Prop.1 correctly regards them as malwares. Furthermore, we also found the 2 benign apps sending IMEI to Unknown servers and resent them to VirusTotal in April 2019. Accordingly, all of the 60 antivirus scanners in VirusTotal do not rise any alarms for them. Although these apps are still regarded as benign ones by VirusTotal, according to our attack model, the apps may be malwares. Thus, we regards the 2 apps as potential malwares. From these results, we conclude that our scheme is promising for detecting new malwares which evade existing solutions.

4.6 False Negative Analysis

Prop.1 misses 55 malwares. After manually checking the level of SSL server certificate and traffic data, we discovered that these malwares do not interact with untrusted servers. This is because we cannot bring out malicious actions of malwares. Most of malwares stealthily perform malicious actions regardless of user's operations. However, some of the malicious activities may be triggered by user's operations and inputs. In our simulation, such actions cannot be triggered because we utilize Monkey tool for triggering malicious actions. Thus, our simulation cannot perfectly emulate the situation like real usage by users. These results reveal the limitation of our simulation. In the future, we must design the mechanism for emulating user's operations in order to precisely bring out malicious actions. More accurate emulation enables to achieve better detection performance.

Table 10 Malware families of malwares in our dataset and their frequency.

Malware Families	Frequency						
smsreg	285	cimsci	4	moavt	1	mobisec	1
autoins	29	utilcode	3	coinminer	1	mobidash	1
jiagu	28	hypay	3	domob	1	marsdaemon	1
revmob	25	wapron	3	mobilespy	1	faveav	1
apptrack	18	umpay	3	kyvu	1	coinge	1
shedun	18	droidrooter	3	gcxgz	1	smsagent	1
dnotua	15	dianjin	2	mgyun	1	wajar	1
tencentprotect	9	pornvideo	2	ewind	1	amhw	1
artemis	9	hiddenads	2	openconnection	1	ganlet	1
skymobi	9	piom	2	uupay	1	ghin	1
waps	8	datacollector	2	remco	1	gexin	1
fakeapp	7	mobwin	2	addisplay	1	appcare	1
wapsx	6	highconfidence	2	lotusid	1	eldorado	1
dowgin	6	tocrenu	2	secneo	1	dogwin	1
hiddad	6	sobot	2	cyfin	1	inmobi	1
ginmaster	6	bips	1	igexin	1	carej	1
airpush	5	kingroot	1	lwzsb	1	vpsdrop	1
smspay	4	youku	1	rotexy	1	Unknown Family	298
kyview	4	smsspy	1	systemmonitor	1		
rootnik	4	plankton	1	wkload	1		
kuguo	4	adwo	1	moplus	1		

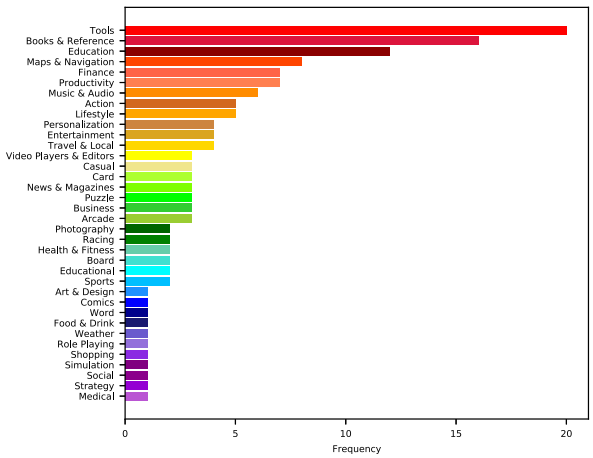

Fig. 4 The categories of benign apps in our dataset and their frequency.

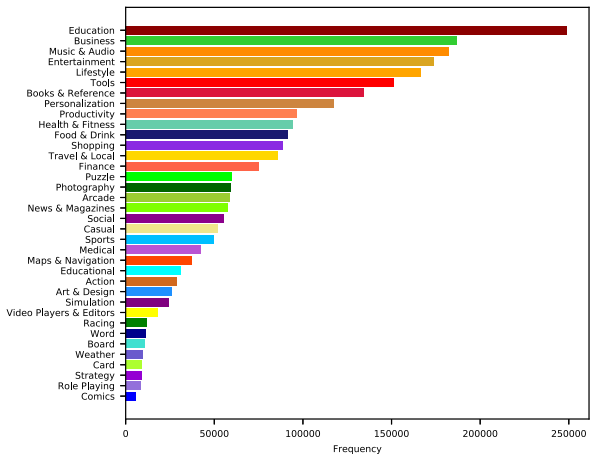

Fig. 5 The distribution of categories in Google Play.

4.7 Discussion of Dataset Bias

In order to clarify a possibility of dataset bias, we inspected categories of benign apps and malwares in our dataset. Out of 801 benign apps, 538 benign ones in our dataset were collected from Google Play. The categories of apps are divided into the 49 types in Google Play. Therefore, we inspected benign apps to clarify the number of categories in our dataset. The categories of 144 benign apps out of 538 benign ones can be identified. The categories of the others cannot be identified due to absence of them in Google Play. Figure 4 shows the categories of benign apps in our dataset and their frequency. As shown in Fig. 4, our dataset contains 36 types of categories of benign apps. The coverage rate of the categories in our dataset to all categories is about 73.5%. Thus, our dataset does not contain only a particular category. From the statistics regarding Google Play categories by AppBrain [21], we identified the distribution of categories in Google Play. Figure 5 shows the distribution of categories in Google Play[†]. As shown in Fig. 5, categories such as "Tools", "Book & Reference" and "Education" are contained in both the top 10 categories of Google play and that of our dataset shown in Fig. 4. This result means that the distribution of categories in our dataset is similar to that of categories in Google Play. Thus, we concluded that there seems to be no possibility of dataset bias regarding categories of benign apps.

On the other hand, Android malwares are classified into categories called "Malware family". In order to identify family name of each malware in our dataset, we used Euphony [22] which is a tool to infer a single family name of a malware. Table 10 shows the names of malware families and their frequency in our dataset. As shown in Table 10, the malwares in our dataset are classified into 80 malware

[†]Please note that we exclude 13 categories which are not in our dataset.

families, and 298 malwares are regarded as "Unknown family". Since Euphony infers malware family on the basis of the detection result by VirusTotal, it cannot classify the apps which cannot be detected by VirusTotal. Although our dataset consists of 80 malware families, "smsreg" accounts for about 32% of our dataset. "smsreg" is a representative malware family which silently collects sensitive data from the device without the user's knowledge or consent. According to the report by Quick Heal [23], since this family accounted for 18% of top 10 malware families detected in first quarter (from January to March) of 2018, it was considerably spread. Even if excluded malwares are considered, the distribution of "smsreg" in our dataset is different from that by the report of Quick Heal. Furthermore, although about 1,264,860 malwares were detected by Quick Heal in first quarter of 2018, VirusShare and Androzoo collected only 28,632 malwares and 15,297 repackaged ones, respectively in 2018. Since the total number of malwares in VirusShare and Androzoo dataset is only 3.4% of the number of malwares detected by Quick Heal, there exists a possibility of dataset bias on VirusShare and Androzoo dataset. Thus, because we obtained malwares from VirusShare and Androzoo in February 2018, it is possible that a particular family such as "smsreg" concentrates in our dataset to some extent. However, we argue that our scheme is useful for detecting malwares because TPR of our scheme is 93.8% as shown in Table 9. This result indicates that our scheme can detect not only "smsreg" but also many types of other malware families.

4.8 Limitation

Our scheme cannot detect the malware that aims at malicious actions such as data tampering or destruction since it does not communicate with none of servers. Because such types of malwares are beyond the scope of our scheme, detecting them by other detection schemes such as API call based one [3] and ICC based one [4] is useful. Focusing on API call and ICC is useful since the malwares which aims at data tampering or destruction must exploit API calls and ICC to access data in user's devices. Furthermore, our scheme cannot also deal with the malware that communicates only with trusted servers such as EV and OV certificate. In this case, attacker's server can be trusted one. However, such situation is rare because EV and OV certificate cannot be obtained by individual. Therefore, we consider that it is difficult for attackers to make their servers trusted ones.

5. Conclusion

In this paper, we have proposed an Android malware detection scheme based on level of SSL server certificate. We utilize SSL server certificate based features. In order to obtain more exact malicious features, we introduce required permission based weight values. The detection performance of our scheme is better than the previous scheme. Our evaluation results show our scheme can deal with encrypted traffic data and the shortcoming of the previous scheme. After manually analyzing misjudged benign apps, we found the apps which send sensitive information to untrusted servers. The analysis results show that our scheme is promising for detecting new malwares.

Acknowledgments

This work is partly supported by the Grant in Aid for Scientific Research (No.17K06440) from Japan Society for Promotion of Science (JSPS).

References

[1] IDC, "Smartphone os market share, 2018 q3." Available: https://www.idc.com/promo/smartphone-market-share/os.
[2] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "Madam: Effective and efficient behavior-based android malware detection and prevention," IEEE Transactions on Dependable and Secure Computing, vol.15, no.1, pp.83–97, 2018.
[3] L. Deshotels, V. Notani, and A. Lakhotia, "Droidlegacy: Automated familial classification of android malware," Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014, p.3, ACM, 2014.
[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.S. Lui, R.T.B. 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] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," IEEE Transactions on Industrial Informatics, vol.14, no.7, pp.3216–3225, 2018.
[7] 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.
[8] 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.
[9] 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.
[10] C. Gañán, O. Cetin, and M. van Eeten, "An empirical analysis of zeus c&c lifetime," Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, pp.97–108, ACM, 2015.
[11] C. Wang, Q. Xu, X. Lin, and S. Liu, "Research on data mining of permissions mode for android malware detection," Cluster Computing, vol.22, pp.13337–13350, 2018.
[12] K. Allix, T.F. Bissyandé, J. Klein, and Y.L. Traon, "Androzoo: Collecting millions of android apps for the research community," Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on, pp.468–471, IEEE, 2016.
[13] "virusshare.com." https://virusshare.com.
[14] "Android studio." https://developer.android.com/studio/index.html.
[15] "Android monkey tool." Available: http://developer.android.com/.
[16] "The tcpdump & libpcap." https://www.tcpdump.org.
[17] "Tshark-the wireshark network analyser." http://www.wireshark.org.
[18] "Openssl cryptgraphy and ssl/tls toolkit." https://www.openssl.org.
[19] L. Breiman, "Random forests," Machine learning, vol.45, no.1,

pp.5–32, 2001.
[20] “Virustotal.com.” Available: https://virusshare.com.
[21] “Appbrain.” Accessed: Sept. 2019. https://www.appbrain.com.
[22] M. Hurier, G. Suarez-Tangil, S.K. Dash, T.F. Bissyandé, Y.L. Traon, J. Klein, and L. Cavallaro, “Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware,” Proceedings of the 14th International Conference on Mining Software Repositories, pp.425–435, IEEE Press, 2017.
[23] “Quick heal.” https://www.quickheal.co.in.

Hiroya Kato was born in Gunma, Japan in 1994. He received his M.S degree from Keio University in 2019. He is a Ph.D. student at Keio University. His research interest is security & privacy for IoT. He is a member of IEICE.

Shuichiro Haruta was born in Saitama, Japan in 1992. He received his M.S. degree from Keio University in 2017. He is a Ph.D. student at Keio University. His research interest is security & privacy for IoT. He was a research associate at Keio University (2017-2018). He is a member of IEICE and IEEE.

Iwao Sasase was born in Osaka, Japan in 1956. He received the B.E., M.E., and D.Eng. degrees in Electrical Engineering from Keio University, Yokohama, Japan, in 1979, 1981 and 1984, respectively. From 1984 to 1986, he was a Post Doctoral Fellow and Lecturer of Electrical Engineering at theUniversity of Ottawa, ON, Canada. He is currently a Professor of Information and Computer Science at Keio University, Yokohama, Japan. His research interests include modulation and coding, broadband mobile and wireless communications, optical communications, communication networks and information theory. He has authored more than 284 journal papers and 430 international conference papers. He granted 45 Ph.D. degrees to his students in the above field. Dr. Sasase received the 1984 IEEE Communications Society (ComSoc) Student Paper Award (Region 10), 1986 Inoue Memorial Young Engineer Award, 1988 Hiroshi Ando Memorial Young EngineerAward, 1988 Shinohara MemorialYoung EngineerAward, 1996 Institute of Electronics, Information, and Communication Engineers (IEICE) of Japan Switching System Technical Group Best Paper Award, and WPMC2008 Best Paper Award. He is now serving as a Vice-President of IEICE. He served as President of the IEICE Communications Society (2012-2014). He was Board of Governors Member-at-Large (2010-2012), Japan Chapter Chair (2011-2012), Director of the Asia Pacific Region (2004-2005), Chair of the Satellite and Space Communications Technical Committee (2000-2002) of IEEE ComSoc., Vice President of the Communications Society (2004-2006), Chair of the Network System Technical Committee (2004-2006), Chair of the Communication System Technical Committee (2002-2004) of the IEICE Communications Society, Director of the Society of Information Theory and Its Applications in Japan (2001-2002). He is Fellowof IEICE, and Senior Member of IEEE, Member of the Information Processing Society of Japan.