PAPER *Special Section on Security, Privacy, Anonymity and Trust in Cyberspace Computing and Communications*

CLAP: Classification of Android PUAs by Similarity of DNS Queries

Mitsuhiro HATADA[†,††a)] ***and*** **Tatsuya MORI**[†b)], ***Members***

SUMMARY This work develops a system called *CLAP* that detects and classifies “potentially unwanted applications” (PUAs) such as adware or remote monitoring tools. Our approach leverages DNS queries made by apps. Using a large sample of Android apps from third-party marketplaces, we first reveal that DNS queries can provide useful information for detection and classification of PUAs. We then show that existing DNS blacklists are limited when performing these tasks. Finally, we demonstrate that the CLAP system performs with high accuracy.

key words: *PUA, PUP, potentially unwanted, DNS query, classification*

1. Introduction

Smartphone users are exposed to threats from “potentially unwanted applications” (PUAs) [1]. Typically, PUA conceals adware, browser toolbars, hacking tools, or remote monitoring tools. Although these may offer certain benefits to the user, they may also trigger “unwanted” behavior such as location tracking. In contrast to malware, PUAs are sometimes installed with the consent of the user, making it challenging to determine whether a PUA should be removed or whether the network access of the software should be blocked. Little attention has been paid to the detection or classification of PUAs, and they have generally been considered a subgroup of bad software.

Recently, however, the number of PUAs that trigger alerts in network security systems has increased to a point where their analysis overwhelms the capacity of incident response teams. Since the team must focus on the malware that presents more critical threats, there is a growing demand for systems that can systematically distinguish PUAs from malware and benign applications.

This study had two goals: firstly, it aims to develop a method for distinguishing between PUAs and malware or benign apps; secondly, it aims to develop a system for classifying different varieties of PUAs on the basis of their behavior. We focused on the Android platform as it allows the user to install apps from third-party sources, making it vulnerable to a wider range of attacks than Apple iOS. Our approach leverages the Domain Name System (DNS) queries that are used by apps. This approach has two advantages. Firstly, as DNS queries can be extracted using symbolic execution or static/dynamic analysis, they are relatively easy to analyze. DNS information is also more robust than other features such as system calls. Secondly, as DNS can be used to control network access, for example by DNS blocking, the extracted information can also be used as a countermeasure against the threats presented by PUAs.

Manuscript received February 8, 2019.
Manuscript revised October 7, 2019.
Manuscript publicized November 11, 2019.

[†]The authors are with Waseda University, Tokyo, 169–8555 Japan.
[††]The author is with NTT Coporation, Tokyo, 100–8116 Japan.
a) E-mail: m.hatada@nsl.cs.waseda.ac.jp
b) E-mail: mori@nsl.cs.waseda.ac.jp

DOI: 10.1587/transinf.2019INP0003

To develop the PUA classification system named *CLAP* and evaluate its performance, we collected a large number of Android apps from third-party marketplaces and built a labeled dataset.

We first investigated the effectiveness of using DNS queries in detecting and classifying PUAs. We used the queries to classify apps into four categories: PUAs, malware, benign apps aimed at the Android, and PUAs aimed at Windows. We also demonstrated the ineffectiveness of existing DNS blacklists for this task. Our methodology was shown to be capable of detecting and classifying PUAs with high levels of accuracy.

Our key contributions are as follows:

- We built the first definitive labeled dataset that we would share with the research community. The dataset includes 5,340 PUAs that account for 237 of distinct varieties. The dataset also includes 5,340 malware and benign apps.
- We reveal that DNS queries are useful metrics for the detection and classification of PUAs and for distinguishing between Android PUAs and Windows PUAs.
- We show that the existing DNS blacklists are limited in detecting or classifying PUAs.
- We present a methodology for using DNS queries to detect and classify PUAs and demonstrate that CLAP performs with high levels of accuracy.

The remainder of this paper is organized as follows. Section 2 gives an overview of the dataset and the methodologies used for measurement and classification. In Sect. 3, we present the measurement and classification results. Section 4 discusses the limitations of the study and makes suggestions for future work. Section 5 summarizes the related studies. Our conclusions are given in Sect. 6.

2. Dataset and Methodology

In this section, we introduce our methodology for distinguishing PUAs, malware, and benign apps, and for classifying PUAs into different varieties. Figure 1 shows the high-

Copyright © 2020 The Institute of Electronics, Information and Communication Engineers

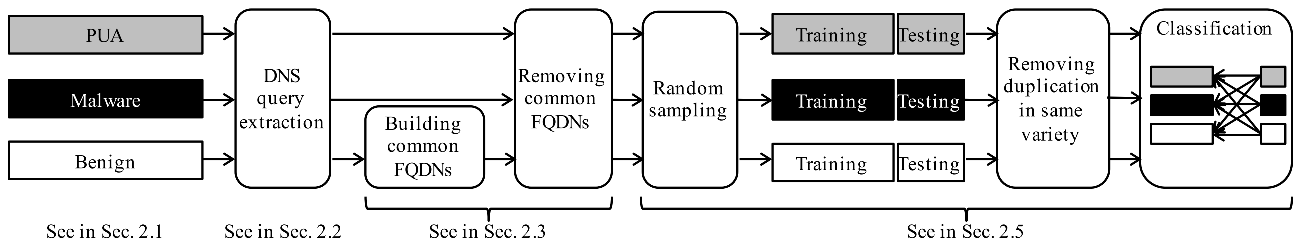


Fig. 1 Overview of CLAP and its evaluation flow.

Table 1 Dataset overview.

# of Android apps	453,687
collection period	Jun. - Aug. 2016
# of PUA samples (varieties)	5,640 (237)
# of malware samples (varieties)	5,640 (393)
# of benign samples	5,640
# of Windows PUA samples (varieties)	5,640 (511)
collection period	Jun. 2016

level overview of CLAP and the evaluation flow of classification; apps collection, measurement of similarity of extracted DNS queries and effectiveness of the existing blacklists, and the classification method with the evaluation process.

2.1 Dataset

Table 1 describes the dataset used in the study. 453,687 apps were retrieved from a number of third-party Android marketplaces[†], which were part of the dataset used in Ref. [2], and were checked using VirusTotal [3]. These marketplaces have huge user bases, and all of the collected apps were free. The focus was placed on third-party marketplaces as they have been shown to harbor approximately 10 times more malicious apps than the official Google Play site [4]. Next, PUAs were identified by searching for certain keywords (“pua”, “pup”, “adware”, “unwanted”, “ ad”, and “/ad”), using VirusTotal. Here, “pup” stands for “potentially unwanted program.” Even with the same detection name, there are samples that perform different behaviors, so it is necessary to prepare multiple samples with the same detection name to improve reliability. In this work, we counted the distinct varieties that had samples larger than 10 in each Anti Virus software (AV) of VirusTotal. As a result, the *ESET-NOD32* with the highest number of PUA varieties was selected for the detection name of PUA and malware in this paper. By randomly selecting 5% in each variety to provide at least 10 samples, we obtained 5,640 samples of 237 varieties. If PUAs, malware, and benign apps can be distinguished, they can be detected in smartphone communication. For this purpose, equivalent samples of malware and benign apps were selected randomly.

We define “malware” as any software that has been flagged by AV but did not contain the PUA keywords and a “benign app” as any software that has not been flagged by any AV. To enable other researchers to perform the replication/extension of our study on Android PUA, we will release the dataset to the research community at https://nsl.cs.waseda.ac.jp/projects/clap/ with verifying user identity. Additional 5,640 Windows PUAs were randomly selected from public malware repositories[††] by searching PUA keywords. They were used to compare Windows PUAs with Android PUAs.

2.2 DNS Query Extraction

To extract the fully qualified domain names (FQDNs) that an Android app may access, a commercial tool was used. This tool uses symbolic execution, giving a broader code coverage, and static analysis of Dalvik byte-code to trace the way in which data are propagated from function to function. From these analyses, we were able to obtain URLs and extract FQDNs from them. In the case of Windows PUAs, packet traces were collected through dynamic analysis using the Cuckoo sandbox [5], and DNS queries were then extracted from the packet traces using *tshark*, a command-line tool of *Wireshark*.

2.3 Common FQDN Removal

Next, we built a list of domain-specific stopwords based on benign apps, which we used this to remove common FQDNs, by applying the document frequency (DF) approach that is widely used in the field of information retrieval. FQDNs with a higher DF were removed from the apps, leaving FQDNs unique to the DNS queries generated by the app. Lists were constructed for each R that identified a portion of all apps, determined by the formula:

$$R = \frac{DF(t)}{N},$$

where t is a candidate common FQDN, $DF(t)$ represents the number of apps with an observed DNS query for t, and N represents the total number of apps. In the case of Windows PUA, we eliminated common domains by matching with the top 10k domains in Alexa [6], which ranks web traffic by

[†]https://www.alandroidnet.com/, http://appvn.com/android, https://www.aptoide.com/, http://shouji.baidu.com/, http://www.blackmart.us/, https://cafebazaar.ir/, http://www.entumovil.cu/downloads/apps, http://www.getjar.com/, http://www.mobogenie.com/, http://www.mobomarket.net/, https://www.uptodown.com/android/, https://m.store.yandex.com/

[††]https://malwr.com/, http://malshare.com/, https://virusshare.com/

volume.

2.4 Measurement

DNS blacklisting is widely applied to detect or block communication by malware. To measure the effectiveness of exiting blacklists, we gathered publcly available blacklists and matched the extracted FQDNs from Section 2.2 with them. Note that the condition of the match is an exact match of FQDN.

The set of domain names used to control applications such as advertisement distribution will be characterized by each PUA. We computed the similarity between the DNS queries of two apps using the Jaccard similarity coefficient. There are several other metrics that can measure the similarity between two finite sets, e.g., Sørensen-Dice coefficient and Szymkiewicz-Simpson coefficient. Compared to these metrics, the Jaccard coefficient is more suitable for our objectives because it captures not only the number of common elements but also the difference in the number of elements,which is essential to our application. Given two sets of DNS queries made by an app, X and Y, the Jaccard similarity coefficient was derived as follows.

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$
$$0 \leq J(X,Y) \leq 1$$

If no common FQDNs were found, $J(X,Y)$ was zero. If all FQDNs were common to the two sets, $J(X,Y)$ was one. The Js between all combinations of PUA samples were computed, and then the average value within each PUA variety was calculated.

2.5 Classification

We randomly chose 10% of the PUAs, malware, and benign apps for testing. The remaining samples were used for training. Samples that did not query any FQDN were excluded from both the testing and training datasets, as they fall outside the scope of CLAP. Duplicated DNS queries within the same variety of PUA were also excluded from the training data. The reduced sample sizes also reduced the complexity of computing J. We then computed J for the testing and training datasets. When the maximum value of J had been obtained, we predicted the category and variety from the training dataset for each testing item. We repeated these steps 10 times and used the average of each metric to improve the reliability of evaluated results. As a comprehensive evaluation metric, we computed the score multiplied by the classification accuracy and the number of test samples. Although detecting *unknown* PUAs is beyond the scope of CLAP, we investigated the applicability of the proposed method for *unknown* PUAs. Specifically, we randomly sampled 90% of PUA varieties to use them for training as known PUAs) and used the rest of the PUA varieties for testing as the *unknown* PUAs. The experiments were performed in the same manner as described in the above steps.

3. Experiments

We next conducted experiments to test the measurement and classification systems introduced in Sects. 3.1 and 3.2.

3.1 Measurement Results

3.1.1 Comparative Evaluation

Table 2 shows the number of distinct FQDNs, the mean number of queries per sample in each category, and the number of distinct FQDNs shared across categories. Android PUAs and malware had a higher mean number of queries per sample than the benign apps. Approximately 30% of the distinct FQDNs were commonly found among Android categories. Only 18 common FQDNs were found between Android PUAs and Windows PUAs, of which 11 were related to Google searches, such as *accounts.google.com*, and seven were related to a browser toolbar, market research, Amazon S3, Facebook, two news websites, and *example.com*. Interestingly, queries for *example.com* that is reserved for use documentation in RFC 6761 [7] were observed in 11 varieties of Android PUA and one variety of Windows PUA. We assumed that this FQDN was used to check accessibility to the Internet or to debug a code in the app.

Table 2 also shows the efficacy of detection in each category by matching the number of FQDNs with those on publicly available blacklists. *EasyList* [8], one of the most popular ad blockers, lists advertising-related keywords and FQDNs, and *ad server* [9] lists only FQDNs. The *malware domain blacklist* [10] is a project that attempts to prevent malware and spyware from installing. Our results suggest that these blacklists have only limited ability to detect Android PUAs and Windows PUAs, or Android malware. Even when a PUA or malware is detected by using these blacklists, it is not an easy task for the incident response team to attach appropriate priority to the infection, because only very limited information is available from detection, for example *attackpage, malware, and phishing* in the list of dnsbh. Note that the 5,124 FQDNs of Android PUAs and 5,259 FQDNs of Android malware contain benign FQDNs, and these benign FQDNs should not be blocked. Of the 96 FQDNs for (3) matched with *EasyList*, 32 were also found in both (1) and (2), and these were probably false positives, for example representing *youtube.com*, *www.msn.com*, or similar sites.

3.1.2 Similarity of DNS Queries

Next, we investigated the similarity of DNS queries in each variety of Android PUA. The results presented here are limited to the case in which $R = 0.0020$ as it achieved the highest score multiplied by the classification accuracy and

Table 2 DNS queries in each category and match with blacklists.

category	mean # of queries per sample	# of distinct FQDN	(1)	(2)	(3)	(4)	EasyList (of 13,620)	ad server (of 2,377)	dns-bh (of 33,302)
(1) Android PUA	34.4	5,124	n/a	1,780	1,563	18	79	10	0
(2) Android malware	37.2	5,259	1,780	n/a	1,498	21	65	7	0
(3) Android benign	23.7	9,502	1,563	1,498	n/a	22	96	10	0
(4) Windows PUA	5.3	635	18	21	22	n/a	5	1	9

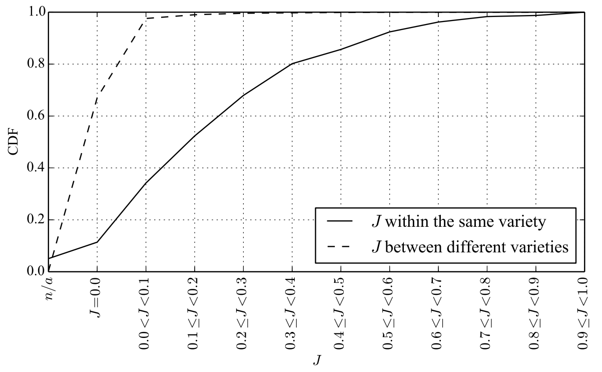
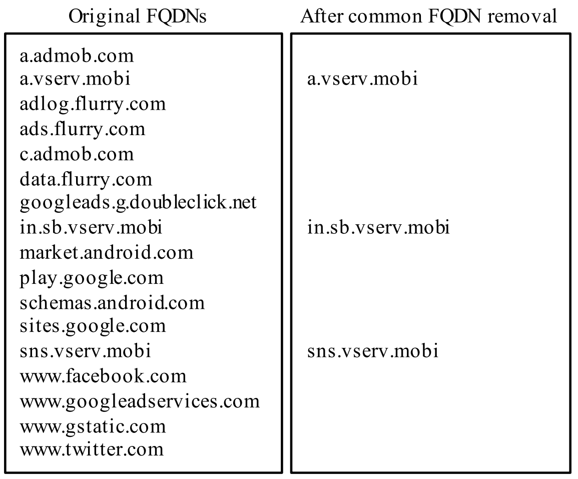


Fig. 2 CDF of J within the same variety and between different varieties. The horizontal axis shows the range of J and *n/a* indicates that no DNS query was extracted from both target apps.



Original FQDNs	After common FQDN removal
a.admob.com	
a.vserv.mobi	a.vserv.mobi
adlog.flurry.com	
ads.flurry.com	
c.admob.com	
data.flurry.com	
googleads.g.doubleclick.net	
in.sb.vserv.mobi	in.sb.vserv.mobi
market.android.com	
play.google.com	
schemas.android.com	
sites.google.com	
sns.vserv.mobi	sns.vserv.mobi
www.facebook.com	
www.googleadservices.com	
www.gstatic.com	
www.twitter.com	

Fig. 3 Example of FQDN extraction: *a variant of Android/Adware. Viser.B*, and common FQDN removal.

the number of test samples between varieties of PUA. Figure 2 shows the relationship between the range of computed Js and the cumulative relative frequency. Note that there were a total of 237 combinations for the same variety and 27,966 for different varieties. As can be seen, 66.8% of the combinations of different varieties of Android PUAs showed similarity of 0.0, and 97.3% showed similarity less than 0.1. In terms of the similarity between the same varieties, only 6.3% of the Android PUA varieties demonstrated similarity of 0.0, 34.2% showed similarity less than 0.1, and 65.8% showed similarity greater than or equal to 0.1. The maximum J value for different varieties was 0.823. There were four varieties for which the computed J for same varieties was greater than 0.823: *AdDisplay.Fictus.F*, *AdDisplay.Fictus.B*, *AdDisplay.AirPush.H*, and *AdDisplay.Fictus.E*. These varieties are completely distinguished from different varieties, and DNS queries are generally useful as metrics for classifying each variety of PUAs.

3.2 Classification Results

3.2.1 Classification of PUAs, Malware, and Benign Apps

Figure 3 gives an example of the FQDNs and the effect of common FQDN removal. Note that the top 100 examples of common FQDNs to be used widely in Android apps when $R = 0.0020$ are listed in Table 3. These were extracted in *a variant of Android/Adware.Viser.B*, which is one of the PUA varieties with the highest classification accuracy. In Fig. 3, three FQDNs remain after common FQDN removal. These FQDNs are common to *vserv.mobi* as the second level domain. The domain is observed in 131 PUA samples, 15 malware samples, and six benign samples. Among these 152 samples, 105 samples are detected as the *Viser* family of PUA. However, since the removed FQDNs such as *admob.com* and *flurry.com* are related to the mobile advertising company, there is a possibility that the common FQDN may include an FQDN that represents the characteristic of a specific PUA. We will discuss the issue of common FQDN removal in Sect. 4. We also provide several examples of FQDNs generated by each variety of PUA after removing common FQDN in Table 4. The examples presented here are limited to the varieties in which ACC is 100%, and the number of samples is greater than 13. Due to space limitation, we omitted "a variant of Android/" from the name of PUA.

Table 5 shows the accuracy of the classification of PUAs, malware, and benign apps. As expected, PUAs were classified with 92.9% accuracy. Malware and benign apps were also accurately classified (91.7% and 96.0%, respectively). By choosing R for common FQDN removal and the sample reduction procedures (Sect. 2.5), the mean number of samples in each category decreased. For example, the mean number of testing samples in PUAs was decreased from 564 to 518.8, which means that 92 % of the testing data remained for evaluation. Figure 4 shows the classification accuracy, the rate of the mean number of testing samples, and the score for PUAs and malware with changes of R. The rate of the mean number of testing samples is the rate of the mean number of testing samples divided by 564, which is 10% of each dataset. As with the results shown in 3.2.2, we found the same tendency, i.e., a decreasing value of R

Table 3 Top 100 examples of Common FQDNs (R = 0.0020).

schemas.android.com	plus.google.com	www.google.com	googleads.g.doubleclick.net	media.admob.com
www.google-analytics.com	ssl.google-analytics.com	www.googleapis.com	play.google.com	www.googletagmanager.com
www.facebook.com	e.admob.com	twitter.com	market.android.com	www.gstatic.com
graph.facebook.com	csi.gstatic.com	m.facebook.com	badad.googleplex.com	schema.org
accounts.google.com	www.paypal.com	api.facebook.com	www.linkedin.com	login.live.com
login.yahoo.com	data.flurry.com	facebook.com	xmlpull.org	www.w3.org
api.twitter.com	www.youtube.com	www.amazon.com	a.admob.com	app-measurement.com
maps.google.com	settings.crashlytics.com	e.crashlytics.com	androidsdk.ads.mp.mydas.mobi	cvt.mydas.mobi
ads.mp.mydas.mobi	millennialmedia.com	api.parse.com	google.com	www.startappexchange.com
i.w.inmobi.com	d1byvlfiet2h9q.cloudfront.net	www.dummy.com	stream.twitter.com	docs.google.com
twitter4j.org	userstream.twitter.com	maps.googleapis.com	images.millennialmedia.com	sitestream.twitter.com
git-wip-us.apache.org	ad.flurry.com	s3.amazonaws.com	gdata.youtube.com	xml.org
sites.google.com	purl.org	goo.gl	ads.flurry.com	adlog.flurry.com
pagead2.googlesyndication.com	m.google.com	schemas.xmlsoap.org	cafebazaar.ir	spreadsheets.google.com
alog.umeng.com	alog.umeng.co	ws.tapjoyads.com	www.googleadservices.com	www.example.com
www.umeng.com	search.twitter.com	my.mobfox.com	www.andromo.com	www.umeng.co
sdk-b.apptornado.com	applift-b.apptornado.com	applift-a.apptornado.com	oc.umeng.com	oc.umeng.co
xml.apache.org	c.admob.com	amazon-adsystem.com	lp.mydas.mobi	java.sun.com
ads.mopub.com	sb-ssl.google.com	r.admob.com	mm.admob.com	crashlytics.com

Table 4 Examples of FQDNs for Android PUAs (R = 0.0020).

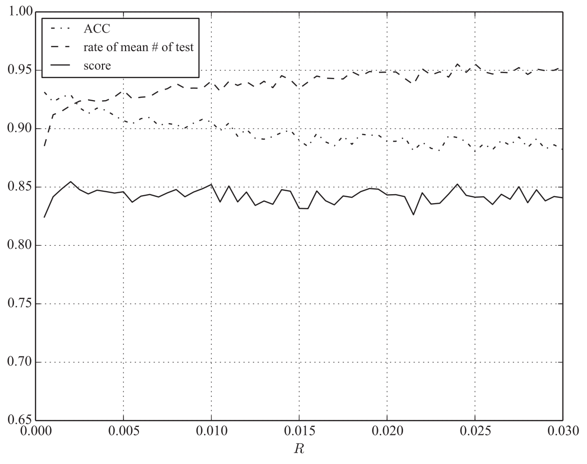
PUA	FQDN		
AdDisplay.Dianru.A potentially unwanted	ads.wapx.cn	www.dianru.com	mob.guohead.com
	app.wapx.cn	www.xabaoyi.com	www.adview.cn
	data.gfan.com	www.guohead.com	sns.whalecloud.com
	api.dianru.com	static.dianru.com	show.lsense.cn
	ads.lmmob.com	cdn1.down.apk.gfan.com	apitest.dianru.com
	m.sc.hiapk.com		
Leadbolt.C potentially unwanted	wx3.lirr.org	shadow01.yumenetworks.com	lirrgcm.jelastic.servint.net
	www.you	lirrmessages.appspot.com	api.sndcdn.com
	cdn1.smartadserver.com	android.bcfads.com	thx.swelen.com
	www.mix.dj	gcp-test.appspot.com	android.mix.dj
	lh5.ggpht.com	www.rateguitar.com	nealo.se
	ads.huntmad.com	ad.leadboltapps.net	ad.leadbolt.net
	www.hulkshare.com	gaurav-gupta.zzl.org	en.dilandau.eu
	www.mta.info	top10songs.com	www.soundcat.ch
	adc.medibaad.com	mobile.smartadserver.com	r.tapit.com
	lirr42.mta.info		
AdDisplay.AppOffer.A potentially unwanted	android.waptw.com	developer.android.com	mnav.fetion.com.cn
	wap.monternet.com	az.damiapk.com	mobile.video.qq.com
	m.appchina.com	mw.app.qq.com	id.godiy8.com
	wap.mpwap.cn	wap.etwap.com	waptw.cn
	client.azrj.cn	m.sc.hiapk.com	www.awapk.com
	az.damiapk.jsp	etwap.com	hdss1ftb.fetion.com.cn
	apkyx.com	apkrj.com	wap.baidu.com
	sc.hiapk.com	www.google.com.hk	m.xzapk.com
	v.qq.com	mobwin.android.com	etwap.cn
	tgwap.com	sns.video.qq.com	update.apksj.com
	a.wap.myapp.com	ring.xgapk.cn	wap.easou.com
	www.damiapk.com	m.baidu.com	down.gfan.com
	az.azrj.cn	az.apksj.cn	m.anzhi.com
	vv.video.qq.com		
AdDisplay.Fictus.F potentially unwanted	app-stats.net2share.com	stats.inappertising.org	serve.vdopia.com
	cfg.inappertising.org	s.net2share.com	
AdDisplay.Viser.A potentially unwanted	apkservice.com	sns.vserv.mobi	cdn1.androidhomebase.com
	api.hungama.com	bucket.homebase-apps.com	winjit.in
	delivery.hungama.com	rq.vserv.mobi	s.vserv.mobi
	a.vserv.mobi	www.apkservice.com	in.sb.vserv.mobi
	exception.homebase-apps.com		
AdDisplay.Viser.C potentially unwanted	i.xx.openx.com	sns.vserv.mobi	smaato-android-sdk.s3.amazonaws.com
	c.vserv.mobi	ads.buzzcity.net	custom.com
	inneractive-assets.s3.amazonaws.com	www.mobimonsterit.com	raw.github.com
	cdn2.inner-active.mobi	www.gameneeti.com	a.vserv.mobi
	www.xercestechnologies.com	www.smaato.com	in.sb.vserv.mobi
a variant of Android/AdDisplay.ADpooh.A	wiyun.com	shun.sinaapp.com	test.adpooh.com
	ad.zhidian3g.cn	d.wiyun.com	www.soso.com
	api.mopay.com	mopaystaging.mindmatics.com	www.google.hk
	n.wiyun.com	wap.miidi.net	com.saubcy.LegoBoxes.Layout

decreases the number of valid testing samples; nevertheless, classification accuracy increases. When the prediction yields the same highest J value with multiple categories, we classified the resulting samples into "PUA or malware," "PUA or benign," "malware or benign," or "PUA, malware, or benign." The predicted rate of "PUA or malware" for both PUA and malware was slightly larger than that of others. However, in this case, this is useful for determining whether a detailed investigation should be conducted with priority over a PUA. In the case of malware, "unclassified" means that the example could not be placed into any category. Although in these cases FQDNs were completely different from those in the training data, we can reduce them by choosing several samples for each variety of malware, as described in Sect. 2.1 for the case of PUAs.

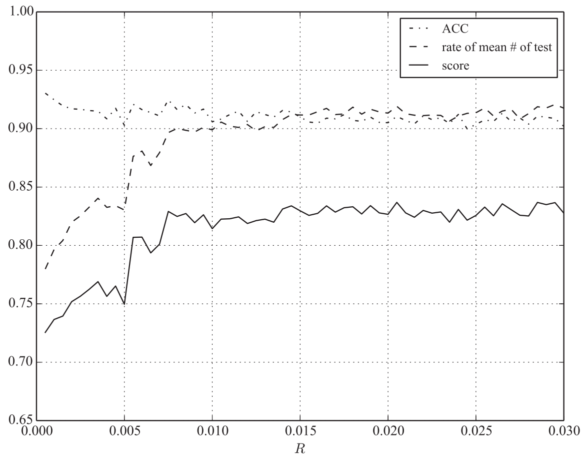
Table 7 gives a breakdown of the top five pairs of PUA

Table 5 Confusion matrix: Classification accuracy of PUAs, malware, and benign apps (R = 0.0020).

category (# of training samples : # of testing samples)	PUA	malware	benign	PUA or malware	PUA or benign	malware or benign	PUA, malware or benign	unclassified
PUA (7991.1 : 518.8)	0.9291	0.0318	0.0066	0.0249	0.0040	0.0015	0.0021	0.0000
malware (7978.6 : 462.4)	0.0368	0.9170	0.0095	0.0182	0.0032	0.0052	0.0013	0.0089
benign (7981.6 : 309.4)	0.0129	0.0168	0.9602	0.0016	0.0039	0.0045	0.0000	0.0000



(a) PUA



(b) malware

Fig. 4 Classification accuracy between categories, rate of mean number of test, and score. The horizontal axis shows the range of R.

that were misclassified as malware in Table 5. The *Domob* family was shown to be a frequent cause of misclassification. *Domob*[†††] is a mobile advertising network based in China that contains a software development kit for incorporation in an app. We detected 1,748 samples of the *Domob* family from both PUAs and malware, 99% of which had been downloaded from the *Baidu* app store, which is one of the main third-party marketplaces in China. The subtle difference in the naming convention was also confirmed by some of the pairs, for example *a variant of Android/Domob.G potentially unwanted* and *probably a variant of Android/Domob.G*. In the pairs where malware was misclassified as a PUA, we observed a similar trend. This is further discussed in Sect. 4. Table 8 provides a breakdown of the top five PUAs that were misclassified as benign in Table 5. The total number of this misclassification was 34, and common FQDNs were observed in each pair, of which 10 were identified as *AirPush*[††††] family, which is a US-based mobile advertising service. Not only the misclassification but also these benign samples could possibly remain undetected by AVs because we found that Symantec Mobile Insight detects a sample as "AppRisk:Generisk" and Alibaba detects another sample as "A.W.Rog.RevMob.C" from VirusTotal (as of August 2017). These cases imply that the signatures of AVs were added later, so these were new varieties at the time of data collection.

Table 6 presents the classification accuracy for unknown PUAs. In this analysis, 147 (10%) varieties of unknown PUAs were randomly sampled 10 times. Compared to the results shown in Table 5, the classification accuracy of these unknown PUAs decreased (55.7%). However, we note that the relaxed classification results that include correct answers ("PUA or malware," "PUA or benign," or "PUA, malware, or benign") are more useful for prioritizing an incident relative to PUAs than benign apps. In that case, accuracy can be considered to be 83.2% according to the calculation result of 0.5565 + 0.1999 + 0.0308 + 0.0443 = 0.8315.

3.2.2 Classification of the Varieties of PUAs

Figure 5 shows the classification accuracy, the rate of the mean number of test samples, and the score for each variety of PUA and malware with changes of R. Table 9 shows the details of the results. It can be seen that the removal of common FQDNs, as described in Sect. 2.3, effectively improved the classification of PUAs. At an R of 0.0020, the accuracy increased by 9.3% over the case without common FQDN removal, achieving the highest accuracy of 85.3%. Note that both the training data and testing data decreased as R decreased. It is evident that decreasing the amount of training data reduces the complexity of classification per sample. Since accuracy is calculated only when a training sample with the highest similarity to the test data is uniquely determined to be the correct answer, it is judged as an error when there are two or more candidates with the equal highest similarity. The term "con. rate" indicates the rate that includes the correct answer in the candidates with same highest sim-

[†††]http://www.domob.cn/
[††††]http://www.airpush.com/

Table 6 Classification accuracy of unknown PUAs ($R = 0.0020$).

(# of training samples : # of testing samples)	PUA	malware	benign	PUA or malware	PUA or benign	malware or benign	PUA, malware or benign	unclassified
unknown PUA (8184.2 : 539.4)	0.5565	0.1144	0.0289	0.1999	0.0308	0.0044	0.0443	0.0208

Table 7 Top 5 examples of PUAs misclassified as malware.

# of misclassification	misclassified PUA in testing data	predicted malware in training data
8	*a variant of Android/**Domob.G** potentially unwanted*	*probably a variant of Android/**Domob.G***
8	*a variant of Android/AdDisplay.AdsWo.A potentially unwanted*	*a variant of Android/**Domob.F***
5	*a variant of Android/AdMogo.A potentially unwanted*	*probably a variant of Android/**Domob.G***
5	*a variant of Android/AdDisplay.AdsWo.A potentially unwanted*	*probably a variant of Android/**Domob.G***
4	*probably a variant of Android/Adware.Youmi.B*	*probably a variant of Android/**Domob.B***

Table 8 Top 5 examples of PUAs misclassified as benign.

# of misclassification	misclassified PUA in testing data
3	*a variant of Android/AdDisplay.AirPush.P potentially unwanted*
3	*a variant of Android/AdDisplay.AirPush.I potentially unwanted*
2	*a variant of Android/Kalfere.A potentially unwanted*
2	*a variant of Android/Adware.Wooboo.D*
2	*a variant of Android/Adload.B potentially unwanted*

ilarity. The value of "con. rate" is less than 2.5%; thus the samples with this "con. rate" mostly constituted different DNS queries and practically caused misclassification. From the 237 varieties of PUA, 234 were selected as testing data after the random sampling and sample reduction described in Sect. 2.5. The accuracy was also computed for each variety of PUA. Of the 234 varieties, 169 (72.2%) had an ACC greater than or equal to 0.7, and seven (3.0%) had ACC values lower than 0.3. We were able to classify a sample within 0.034 s using Python 2.7 over Ubuntu 14.04.5 LTS running on a Dell PowerEdge R210 Ⅱ (CPU: 2.3 GHz Intel Xeon E3-1220Lv2, Memory: 32 GB 1333 MHz UDIMM).

When the highest score was 0.799, the accuracy of 86.9% was achieved at $R = 0.0205$ for 261 of the 393 varieties of malware covered by the testing data. Compared to PUA, ACC is high irrespective of R. There may be multiple FQDNs specific to malware. CLAP showed the possibility that Android malware can also be classified accurately using the same methodology.

4. Discussion

In this section, we discuss several limitations and applications of our study and suggest future research directions.

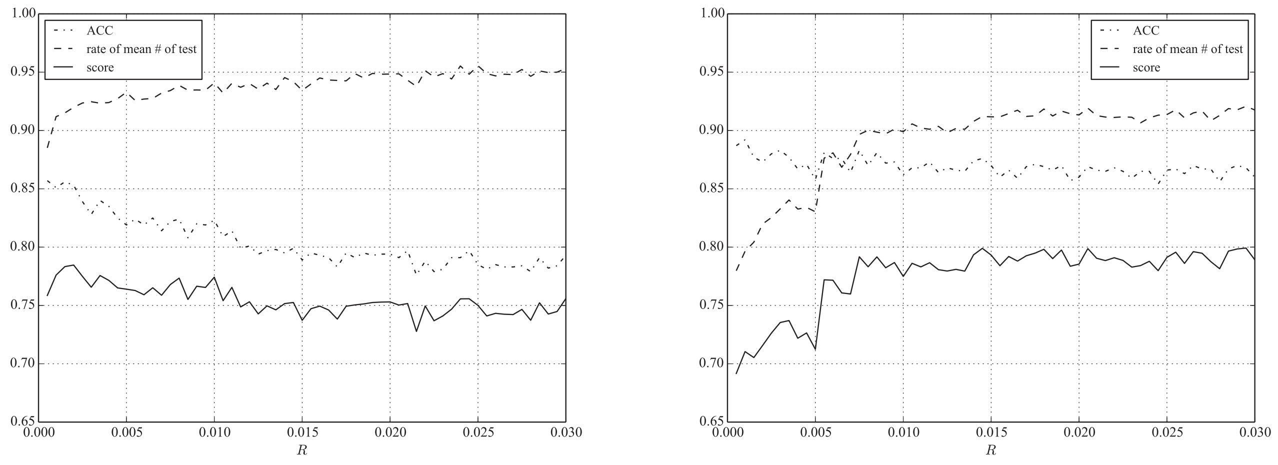
4.1 Limitations

App platform: This study focused only on the Android platform, with a partial analysis of Windows. Other platforms, including Apple iOS and Windows Mobile, were not considered. To extend the coverage, apps will need to be collected from a full range of stores and analyzed to extract the DNS queries specific to each platform. We believe that the approach presented in this paper can be applied to these other platforms.

PUA selection: In this study, we selected PUA varieties based on keywords in the name that could be detected by AV. This is clearly vendor-specific. AVClass [11] is a technique that allows a common name to be assigned by summarizing the names detected by multiple AVs. As the accuracy of classification entirely depends on the use of appropriate labels, this technique may be used to improve the methods presented in this paper.

FQDN extraction: As described in Sect. 2.2, we extracted FQDNs from all possible URLs accessed by the app. However, the app will not necessarily access all of these URLs in user operation or background execution. If only those FQDNs actually accessed by an app through dynamic analysis are considered, the accuracy of classification may decrease. This is also a challenge in dynamic analysis with an automated user operation. Even if FQDNs can be extracted properly, the proposed method can be evaded by a PUA using a domain generation algorithm (DGA). Since various types of DGAs are involved, it is difficult to correctly determine an FQDN generated by a DGA. When checking all extracted FQDNs from Android PUAs and malware with two feeds [12], [13] that typically cover DGA families, DGA domains did not exist. If DGA domains are used by Android PUAs or malware, classification accuracy will be affected. By replacing a part of a domain generated by a DGA with static characters (e.g., from *vqugxwskcupgevv.example.com* to ***15length-dga**.example.com*), an improvement in accuracy may be expected.

Common FQDN removal: Since the number of test apps will decrease as R decreases, there is a trade-off between PUA coverage and classification accuracy (between PUA and others, and among varieties of PUA). As discussed in Sect. 3.2, common FQDN removal may eliminate FQDNs that are specific to particular varieties of PUA. To classify PUAs or malware more accurately, building a list of common FQDNs that use more benign apps, including apps on the official Google Play, may be needed. We used DF



(a) PUA

(b) malware

Fig. 5 Classification accuracy between varieties, rate of mean number of test, and score. The horizontal axis shows the range of *R*.

Table 9 Detail of classification accuracy between varieties.

(a) PUA

R (# of removal FQDN)	ACC	mean *J*	max. *J*	min. *J*	med. *J*	std. *J*	con. rate	mean # of train	mean # of test	mean time [s]	score
0.0010 (685)	0.851	0.635	0.968	0.076	0.662	0.182	0.024	7606.4	514.3	0.032	0.776
0.0020 (373)	**0.853**	**0.639**	**0.970**	**0.070**	**0.654**	**0.180**	**0.025**	**7991.1**	**518.8**	**0.034**	**0.785**
0.0030 (266)	0.828	0.640	0.969	0.093	0.657	0.179	0.015	8151.8	521.5	0.035	0.766
0.0040 (210)	0.835	0.648	0.972	0.088	0.667	0.182	0.014	8245.2	521.1	0.037	0.771
0.0050 (175)	0.819	0.650	0.970	0.109	0.667	0.180	0.019	8292.6	526.1	0.037	0.764
0.0060 (155)	0.819	0.644	0.969	0.098	0.664	0.182	0.015	8339.3	522.8	0.038	0.759
0.0070 (139)	0.814	0.655	0.972	0.077	0.667	0.179	0.023	8435.2	525.7	0.039	0.759
0.0080 (115)	0.824	0.656	0.973	0.109	0.667	0.178	0.024	8543.5	529.4	0.039	0.773
0.0090 (102)	0.820	0.654	0.972	0.108	0.667	0.183	0.019	8587.0	527.2	0.040	0.766
0.0100 (91)	0.823	0.658	0.973	0.085	0.667	0.186	0.018	8631.2	530.6	0.040	0.774
0.0150 (62)	0.789	0.671	0.972	0.116	0.667	0.184	0.015	8806.6	527.0	0.044	0.737
0.0200 (53)	0.794	0.666	0.972	0.088	0.674	0.187	0.018	8861.7	534.9	0.043	0.753
0.0205 (52)	0.791	0.669	0.972	0.093	0.674	0.184	0.014	8902.8	535.0	0.043	0.750
0.0250 (42)	0.785	0.680	0.973	0.106	0.687	0.183	0.012	9004.1	538.9	0.046	0.750
0.0300 (38)	0.793	0.679	0.973	0.107	0.691	0.184	0.008	9018.0	537.6	0.045	0.756
w/o removal (0)	0.760	0.735	0.975	0.155	0.757	0.169	0.004	9978.6	545.2	0.060	0.735

(b) malware

R (# of removal FQDN)	ACC	mean *J*	max. *J*	min. *J*	med. *J*	std. *J*	con. rate	mean # of train	mean # of test	mean time [s]	score
0.0010 (685)	0.892	0.669	0.966	0.080	0.698	0.185	0.015	7597.5	449.2	0.031	0.710
0.0020 (373)	0.873	0.674	0.967	0.055	0.703	0.185	0.023	7978.6	462.4	0.033	0.716
0.0030 (266)	0.883	0.684	0.965	0.073	0.713	0.180	0.020	8126.0	469.7	0.034	0.735
0.0040 (210)	0.867	0.683	0.968	0.064	0.705	0.179	0.008	8219.7	469.6	0.036	0.722
0.0050 (175)	0.858	0.692	0.967	0.069	0.714	0.182	0.014	8279.5	468.3	0.036	0.712
0.0060 (155)	0.876	0.683	0.967	0.070	0.700	0.180	0.006	8356.1	496.8	0.037	0.772
0.0070 (139)	0.864	0.690	0.969	0.086	0.714	0.178	0.010	8442.4	496.0	0.038	0.760
0.0080 (115)	0.870	0.710	0.970	0.077	0.750	0.166	0.007	8527.4	507.7	0.039	0.783
0.0090 (102)	0.872	0.718	0.968	0.085	0.742	0.156	0.005	8588.3	506.0	0.039	0.782
0.0100 (91)	0.862	0.722	0.970	0.053	0.750	0.161	0.007	8636.7	507.0	0.040	0.775
0.0150 (62)	0.870	0.730	0.970	0.107	0.769	0.158	0.001	8829.1	514.2	0.043	0.793
0.0200 (53)	0.860	0.727	0.969	0.074	0.766	0.163	0.007	8899.0	515.1	0.043	0.785
0.0205 (52)	**0.869**	**0.727**	**0.969**	**0.075**	**0.772**	**0.165**	**0.005**	**8900.7**	**518.4**	**0.042**	**0.799**
0.0250 (42)	0.866	0.740	0.968	0.083	0.778	0.158	0.005	8964.5	515.3	0.044	0.791
0.0300 (38)	0.860	0.737	0.971	0.066	0.771	0.160	0.004	9007.3	517.5	0.044	0.789
w/o removal (0)	0.854	0.798	0.973	0.118	0.840	0.151	0.001	9934.0	522.5	0.061	0.791

to build a list of common FQDNs. However, other algorithms can also be used to build an appropriate list of common FQDNs. In addition, Google URL shortener *goo.gl* is listed in Table 3. This FQDN in the dataset is utilized by 12 samples in 11 PUA varieties. When such services are widely used, this type of FQDN will be added to common FQDNs, which makes it impossible to calculate the similarity between apps. In such a case, the original FQDN can be extracted by accessing a shortened URL via dynamic analysis.

4.2 Applications

To detect PUA infection via network monitoring with CLAP, it is necessary to separate each host based on the source IP address of DNS queries. We can then apply CLAP to a set of DNS queries that are divided in a certain period of time. Although DNS queries are generated by benign apps, malware, PUA, and the browser used by the user, CLAP can provide a list of candidates, indicating that the host may be infected. DNS analysis techniques for detecting malicious domains [14] and detecting C&C domains [15] have previously been studied. CLAP focused only on the similarity of FQDNs accessed by apps to distinguish Android PUAs from others and classify varieties of Android PUAs. Identifying malicious domains is outside the scope of this study. However, we may leverage these approaches and features used in previous studies to be applicable for detecting PUA infections. As another use case, when the incident response team faces an alert such as "malicious DNS query." from the network security system, the corresponding FQDN described in the alert details can be inquired into CLAP to identify malware or PUA. If it is PUA, the team can prioritize other alerts, and the team can efficiently triage incidents. Collecting and analyzing apps and accumulating data continuously are crucial for maintaining the performance of CLAP. Compared to the Windows malware that is hidden so as not to be removed from the compromised server, Android apps can be downloaded from marketplaces so collecting apps is not a serious barrier.

5. Related Studies

The attention given to PUAs has increased over recent years. Previous studies have focused on detection [16], [17] or classification [18]–[21] of Android malware and recently addressed the analysis of PUAs [22]–[27]. Although it is difficult to simply compare these studies, our research focuses on Android PUAs and is the first one that aims to measure and classify using a larger dataset than previous studies. Furthermore, compared to previous studies using various features, we achieve the almost same level of classification accuracy by using only DNS queries that are a feature easy to monitor and control network access of compromised hosts. Here, we review some recent studies of Android malware and PUAs.

Detection of Android malware: MADAM [16] proposed a host-based Android malware detection system that utilized different levels of features, including app metadata, user activity, SMS transmission, and system calls. Seven malicious behavioral patterns were defined, including those of PUAs, and correlations were made between patterns and features. More than 96% of malicious apps were detected from 2,800 apps, with a low false positive rate. CREDROID [17] was designed to detect Android malware using VirusTotal to score apk files against the reputation of the URL being accessed by the app, the data being sent out, and the communication protocol. The reputation of the URL was derived from the Web of Trust (https://www.mywot.com/). By observing 1,260 samples from 49 families, it was found that approximately 63% of apps generated network traffic.

Classification of Android malware: DroidScribe [18] developed a framework for the multi-class classification of runtime behavior of Android malware. Feature sets were categorized into network access, file access, binder methods, and file execution. With their new approach to refine SVM classification using Conformal Prediction, the authors of DroidScribe achieved 94% classification accuracy using 5,246 Android malware samples. Chen et al. [19] investigated the robustness of several malware classifiers using different machine learning methods. L1-Regularized Linear Regression was the most robust method, and semantics-based features such as "reachables," "happen-befores," and "unwanted behaviors," improved the robustness of malware classifiers. DroidSieve [20] was a fast, scalable, and accurate system for Android malware detection and family identification. A novel set of features for static detection comprises embedded assets and native code. For both malware detection and family identification, an accuracy of over 99% was achieved in this research. Monet [21] was a lightweight in-device malware detection system for Android devices. A signature for malware detection was implemented to represent logic structures and runtime behaviors of an application. A total of 3,723 malware samples with 500 benign apps were used in the evaluation. The results indicated that an accuracy of around 99% was achieved in detecting malware varieties, with approximately 7% performance overhead and approximately 3% battery overhead.

Taxonomy of PUAs: Zhou et al. [22] systematically characterized the same dataset used in [17] and showed that 86% were repackaged legitimate apps, while 93% exhibited the same capabilities as a bot. They reported a detection rate by four AVs of 79.6% in the best case and 20.2% in the worst case. This 2012 study identified the need for better AVs for mobile apps. Svajcer et al. [23] introduced a structured PUA taxonomy for mobile apps that defined the PUA classification criteria to be used by security vendors and testing organizations.

Distribution of PUAs: Kotzias et al. [24] analyzed the large-scale distribution of PUAs through pay-per-install (PPI) services. They reported that 54% of 3.9 million hosts had installed PUAs, that 65% of PUAs installed further PUAs, and that 25% PUAs were distributed through 23 PPI services. Thomas et al. [25] investigated four major PPI services that distributed 160 software each week. Of these, 59% were identified as "unwanted" by at least one AV. Ransomware behavior, and the ability to evade AVs and virtual environments were observed in these PUAs. They also reported 3.5 million alerts from Google Safe Browsing when making PPI downloads.

Analysis of PUAs: Andow et al. [26] developed lightweight heuristics for triage of mobile grayware that leverages text analytics and static program analysis. Nine categories of grayware were defined on the basis of the installation and runtime behavior. A large-scale study of grayware on Google Play was conducted, and the effectiveness of triage was demonstrated by reducing from 1 million apps to tens of apps. Chen et al. [27] analyzed potentially harmful libraries (PhaLib) across Android and iOS platforms. These were repackaged as legitimate libraries and propagated as malware. The approach was used to map, Android PhaLib to iOS libraries, and to correlate suspicious behavior. They identified 117 Android PhaLibs in the search of over 1.3 million Android apps, and 46 PhaLibs in 140,000 iOS apps.

6. Conclusion

In this work, we developed the CLAP system that aims to classify PUAs based only on the use of DNS queries and built the first labeled dataset of Android PUAs to share with the research community. We first extracted FQDNs from the DNS queries generated by the app and built a list of common FQDNs based on DF. This was then applied to the set of FQDNs extracted from a PUA as domain-specific stopwords. A Jaccard similarity coefficient of the set of FQDNs was computed between all PUA pairs. Significant differences were found between the DNS queries of Android PUAs and Windows PUAs. PUA pairs of the same variety were found to have a much higher similarity than those of different types. The study suggested that currently available blacklists were limited for detection and classification. From these results, we predicted the category and variety from the training dataset for each testing datum when the highest similarity value had been obtained. To reduce computational complexity, duplicated sets of FQDNs within the same variety were removed. This allowed us to easily distinguish Android PUAs, malware, and benign apps with approximately over 92% accuracy. More than 230 varieties of PUA are classified correctly with 85.3% accuracy, with a computational time of 0.034 s per app. The study provides evidence that Android malware can also be accurately classified using the same methodology.

Acknowledgments

A part of this work was supported by JSPS Grant-in-Aid for Scientific Research B, Grant Number 16H02832. We thank Mr. Yuta Ishii and Mr. Sho Mizuno for collecting valuable datasets.

References

[1] C. Pickard and S. Miladinov, "Rogue software: Protection against potentially unwanted applications," Proc. 2012 7th International Conference on Malicious and Unwanted Software, pp.1–8, Oct. 2012.

[2] Y. Ishii, T. Watanabe, M. Akiyama, and T. Mori, "Clone or Relative?: Understanding the Origins of Similar Android Apps," Proc. 2016 ACM on International Workshop on Security And Privacy Analytics (IWSPA '16), pp.25–32, March 2016.

[3] "VirusTotal - Free Online Virus, Malware and URL Scanner." https://www.virustotal.com/ (accessed 2019-02-06).

[4] Y. Kikuchi, H. Mori, H. Nakano, K. Yoshioka, T. Matsumoto, and M. van Eeten, "Evaluating Malware Mitigation by Android Market Operators," 9th Workshop on Cyber Security Experimentation and Test (CSET 16), pp.1–8, Aug. 2016.

[5] "Cuckoo Sandbox: Automated Malware Analysis." https://cuckoosandbox.org/ (accessed 2019-02-06).

[6] "Alexa Top Sites." https://www.alexa.com/topsites (accessed 2019-02-06).

[7] https://tools.ietf.org/html/rfc6761 (accessed 2019-02-06).

[8] "EasyList." https://easylist-downloads.adblockplus.org/easylist.txt (accessed 2019-02-06).

[9] "Blocking with ad server and tracking server hostnames." https://pgl.yoyo.org/as (accessed 2019-02-06).

[10] "DNS-BH - Malware Domain Blacklist." http://www.malwaredomains.com/ (accessed 2019-02-06).

[11] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVClass: A Tool for Massive Malware Labeling," Research in Attacks, Intrusions, and Defenses, Lecture Notes in Computer Science, vol.9854, pp.230–253, Springer International Publishing, Cham, 2016.

[12] "DGA Domain List." http://data.netlab.360.com/feeds/dga/dga.txt (accessed 2019-02-06).

[13] "Domain feed of known DGA domains." http://osint.bambenekconsulting.com/feeds/dga-feed.txt (accessed 2019-02-06).

[14] L. Bilge, E. Kirda, C. Kruegel, and Balduzzi, "Exposure: Finding malicious domains using passive dns analysis," Proc. 18th Annual Network and Distributed System Security Symposium (NDSS '11), pp.1–17, Feb. 2011.

[15] R. Villamarín-Salomón and J.C. Brustoloni, "Bayesian bot detection based on dns traffic similarity," Proc. 2009 ACM symposium on Applied Computing (SAC '09), pp.2035–2041, March 2009.

[16] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Trans. Dependable and Secure Comput., vol.15, no.1, pp.83–97, 2018.

[17] J. Malik and R. Kaushal, "CREDROID: Android Malware Detection by Network Traffic Analysis," Proc. 1st ACM Workshop on Privacy-Aware Mobile Computing (PAMCO '16), pp.28–36, July 2016.

[18] S.K. Dash, G. Suarez-Tangil, S. Khan, K. Tam, M. Ahmadi, J. Kinder, and L. Cavallaro, "DroidScribe: Classifying android malware based on runtime behavior," Proc. 2016 IEEE Security and Privacy Workshops (SPW 2016), pp.252–261, May 2016.

[19] W. Chen, D. Aspinall, A.D. Gordon, C. Sutton, and I. Muttik, "More semantics more robust: Improving android malware classifiers," Proc. 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '16), pp.147–158, July 2016.

[20] G. Suarez-Tangil, S.K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "DroidSieve: Fast and accurate classification of obfuscated android malware," Proc. 7th ACM on Conference on Data and Application Security and Privacy (CODASPY '17), pp.309–320, March 2017.

[21] 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 Trans. Inform. Forensic Secur., vol.12, no.5, pp.1103–1112, May 2017.

[22] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," Proc. 2012 IEEE Symposium on Security and Privacy (S&P 2012), pp.95–109, May 2012.

[23] V. Svajcer and S. McDonald, "Classifying PUAs in the Mobile Environment," Virus Bulletin Conference, Oct. 2013.

[24] P. Kotzias, L. Bilge, and J. Caballero, "Measuring PUP Prevalence and PUP Distribution through Pay-Per-Install Services," Proc. 25th USENIX Security Symposium (USENIX Security 16), pp.739–756, Aug. 2016.

[25] K. Thomas, J.A.E. Crespo, R. Rasti, J.M. Picod, C. Phillips, M.A. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M.A. Courteau, L. Ballard, R. Shield, N. Jagpal, M.A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, "Investigating Commercial Pay-Per-Install and the Distribution of Unwanted Software," Proc. 25th USENIX Security Symposium (USENIX Security 16), pp.721–739, Aug. 2016.

[26] B. Andow, A. Nadkarni, B. Bassett, W. Enck, and T. Xie, "A Study of Grayware on Google Play," Proc. 2016 IEEE Security and Privacy Workshops (SPW 2016), pp.224–233, May 2016.

[27] K. Chen, X. Wang, Y. Chen, P. Wang, Y. Lee, X. Wang, B. Ma, A. Wang, Y. Zhang, and W. Zou, "Following Devil's Footprints:

Cross-Platform Analysis of Potentially Harmful Libraries on Android and iOS," Proc. 2016 IEEE Symposium on Security and Privacy (S&P 2016), pp.357–376, May 2016.

Mitsuhiro Hatada is currently a manager at NTT Corporation, and an adjunct researcher at Waseda University. He received his B.E. and M.E. degrees in computer science and engineering, and Ph.D in engineering from the Waseda University in 2001, 2003 and 2018, respectively. He joined NTT Communications Corporation in 2003 and has been engaged in the R&D of applied security on anti-malware and threat intelligence. He is a member of IPSJ and IEICE.

Tatsuya Mori is currently a professor at Waseda University, Tokyo, Japan. He received B.E. and M.E. degrees in applied physics, and Ph.D. degree in information science from the Waseda University, in 1997, 1999 and 2005, respectively. He joined NTT lab in 1999. Since then, he has been engaged in the research of measurement and analysis of networks and cyber security. From Mar 2007 to Mar 2008, he was a visiting researcher at the University of Wisconsin-Madison. He received Telecom System Technology Award from TAF in 2010 and Best Paper Awards from IEICE and IEEE/ACM COMSNETS in 2009 and 2010, respectively. Dr. Mori is a member of ACM, IEEE, IEICE, and IPSJ.