Detecting and Classifying Android PUAs by similarity of DNS queries

Mitsuhiro Hatada*† and Tatsuya Mori*
* *Waseda University*
3-4-1 Okubo Shinjuku, Tokyo, Japan 169-8555
Email: {m.hatada, mori}@nsl.cs.waseda.ac.jp
† *NTT Communications Corporation*
Gran Park Tower 16F, 3-4-1 Shibaura Minato, Tokyo, Japan 108-8118
Email: m.hatada@ntt.com

***Abstract*—This work develops a method of detecting and classifying "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 the detection and classification of PUAs. Next, we show that existing DNS blacklists are ineffective to perform these tasks. Finally, we demonstrate that our methodology performed with high accuracy.**

1. Introduction

Smartphone users are exposed to threats from "potentially unwanted applications" (PUAs) [1]. Typically, PUA conceals adware, browser toolbar, hacking tool, or remote monitoring tool. While these may offer certain benefits to the user, they may also trigger "unwanted" behavior such as location tracking. In contrast with 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 to be a subgroup of bad software.

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

This study had two goals: to develop a method of distinguishing between PUAs and malware or benign apps and 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 the iPhone. Our approach leverages the Domain Name System (DNS) queries that are used by apps. This has two advantages. First, 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. Second, 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.

To develop the PUA detection methodology and evaluate its performance, we collected a large number of Android apps from third-party marketplaces. 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, and 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 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 ineffective in detecting or classifying PUAs.
- We present a methodology for using DNS queries to detect and classify PUAs and demonstrate that it 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 Section 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 Section 6.

2. Dataset and Methodology

In this section, we introduce our methodology for distinguishing between PUAs, malware, and benign apps, and for classifying PUAs into different varieties.

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 samples for building common FQDN	21,838
# of Windows PUA samples (varieties)	5,640 (511)
collection period	Jun. 2016

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[1], and were checked using Virus Total [2]. The focus was placed on third-party marketplaces as they have been shown to harbor approximately ten times more malicious apps than the official Google Play site [3]. Next, PUAs were identified by searching for certain keywords ("pua", "pup", "adware", "unwanted", " ad", and "/ad"), using anti-virus software (AV). Here, "pup" means "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 which had samples larger than 10 in each AV. The *ESET-NOD32* with the highest number of PUA varieties was then selected. By randomly selecting 5% in each variety so as to be at least 10 samples, we obtained 5,640 samples of 237 varieties. If PUAs, malware, and benign apps can be distinguished, it could be applied to detect them from smartphone communication. For this purpose, equivalent samples of malware and benign apps were randomly selected. We defined "malware" as any software that was flagged by the AV but did not contain the PUA keywords, and a "benign app" as any software that was not flagged by any AV. Common DNS queries used by Android apps can also be used to remove noise for measurement of similarity and for classification. We randomly chose 21,838 samples as 5% of samples from all apps excluding PUAs, malware, and benign apps that were already selected. A further 5,640 Windows PUAs were randomly selected from public malware repositories[2] by searching PUA keywords. There 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 that data is propagated from function to function. From these analyses, we were able to obtain URLs and to extract FQDNs from them. In the case of Windows PUAs, packet traces were collected through dynamic analysis, using the Cuckoo sandbox [4], and DNS queries were then extracted from the packet traces using *tshark*, a command-line tool of *Wireshark*.

1. 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/
2. https://malwr.com/, http://malshare.com/, https://virusshare.com/

2.3. Common FQDN removal

Next, we built a list of domain-specific stopwords, and 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 below, 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 eliminate common domains by matching with the top 10k domains in Alexa [5], which ranks web traffic by volume.

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

2.4. Measurement of similarity

We computed the similarity between the DNS queries of two apps using the Jaccard similarity coefficient. Given two sets of DNS queries made by an app, X and Y, the Jaccard similarity coefficient was derived as follows.

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

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, then the average value within each PUA variety was calculated.

2.5. Classification

Figure 1 shows the evaluation flow of 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 our classification system. Duplicated DNS queries within the same variety of PUA were also excluded from the testing data. The reduced sample sizes

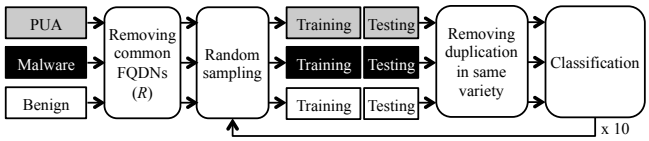

Figure 1. Evaluation flow of classification.

also reduced the complexity of computing J. We then computed J for each testing and training dataset. 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 ten times and used the average of each metric.

3. Experiments

We next conducted experiments to test the measurement and classification systems introduced in Sections 3.1 and 3.2. The results presented here are limited to the case in which $R = 0.01$, as this achieved the highest classification accuracy between varieties of PUA.

3.1. Measurement results

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. The Android PUAs and malware had a higher mean number of queries per sample than the benign apps. Approximately 30% of the distinct FQDNs were found among the Android categories. Only eight common FQDNs were found between Android PUAs and Windows PUAs, of which three were related to Google searches and four to a browser toolbar, to market research, and to two news sites. Interestingly, queries for *example.com* 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 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. *adblock* [6], one of the most popular ad blockers, listing advertising-related keywords and FQDNs, while *ad server* [7] lists only FQDNs. The *malware domain blacklist* [8] is a project that attempts to prevent malware and spyware from installing. Our results suggest that these blacklists have only a 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 no easy task for the incident response team to attach an appropriate priority to the infection, because only very limited information is available from the detection, for example *attackpage, malware, and phishing* in the list of dns-bh. Of the 87 FQDNs for (3) matched with *adblock*, 23 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.

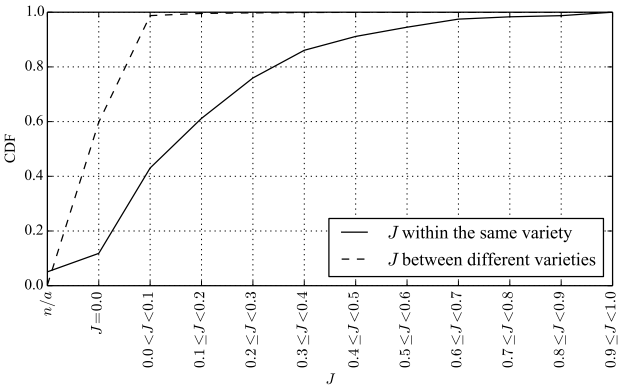

Figure 2. CDF of J within the same variety and between different varieties. The horizontal axis shows the range of J and *n/a* indicates that no DNS query was extracted from both target apps.

Next, we investigated the similarity of DNS queries in each variety of Android 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. It can be seen that, when $J < 0.1$, 59.6% of the combinations for different varieties of Android PUA were not completely similar, and 98.5% were not very similar. When considering the similarity between the same variety, only 6.8% of the Android PUA varieties were not self-similar, but this rose to 57.0% when $J \geq 0.1$. The maximum J value for different varieties was 0.743. There were five varieties for which the computed J was greater than 0.743: *AdDisplay.AirPush.H*, *AdDisplay.Dowgin.B*, *AdDisplay.Fictus.B*, *AdDisplay.Fictus.E*, and *AdDisplay.Fictus.F*.

3.2. Classification results

Table 3 shows the accuracy of classification for PUAs, malware, and benign apps. As expected, PUAs were classified with 89.3% accuracy. Malware and benign apps were also accurately classified, at 93.2% and 97.5%. In the case of malware, "unknown" means that the example could not be placed into any category. Although in these cases the 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 Section 2.1 for the case of PUAs.

Table 4 gives a breakdown of the top-5 pairs of PUA that were misclassified as malware in Table 3(a). The *Domob* family was shown to be a frequent cause of misclassification. *Domob*[3] 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*

TABLE 2. DNS QUERIES IN EACH CATEGORY AND MATCH WITH BLACKLISTS.

category	mean # of queries per sample	# of distinct FQDN	(1)	(2)	(3)	(4)	adblock (of 13,620)	ad server (of 2,377)	dns-bh (of 33,302)
(1) Android PUA	14.2	4,990	n/a	1,646	1,433	11	70	9	0
(2) Android malware	15.9	5,125	1,646	n/a	1,368	11	56	6	0
(3) Android benign	8.6	9,372	1,433	1,368	n/a	12	87	9	0
(4) Windows PUA	5.3	635	11	11	12	n/a	5	1	9

TABLE 3. CLASSIFICATION ACCURACY OF PUAS, MALWARE, AND BENIGN APPS.

(a) $R = 0.08$ (best ACC for PUAs).

	PUA	malware	benign	unknown
PUA	**0.893**	0.073	0.034	0.000
malware	0.046	0.917	0.032	0.005
benign	0.015	0.019	0.967	0.000

(b) $R = 0.02$ (best ACC for malware).

	PUA	malware	benign	unknown
PUA	0.871	0.086	0.043	0.000
malware	0.036	**0.932**	0.026	0.006
benign	0.013	0.015	0.971	0.000

(c) $R = 0.06$ (best ACC for benign).

	PUA	malware	benign	unknown
PUA	0.874	0.084	0.042	0.000
malware	0.044	0.921	0.031	0.004
benign	0.011	0.014	**0.975**	0.000

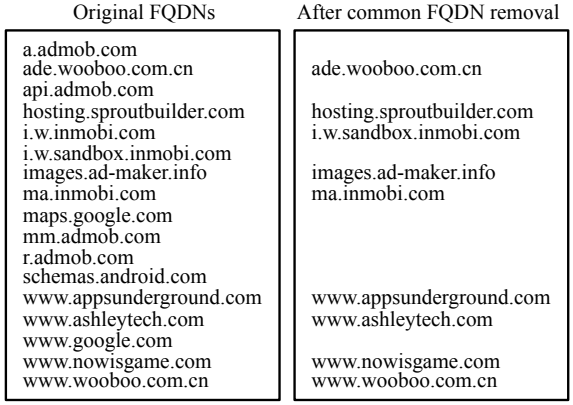

Figure 3. Example of FQDN extraction: *probably a variant of Android/Adware.Wooboo.C*, and common FQDN removal.

unwanted and *probably a variant of Android/Domob.G*. In the pairs where malware was misclassified as a PUA, we found a similar trend. This is discussed further in Section 4.

Table 5 shows the classification accuracy for each variety of PUA. It can be seen that the removal of common FQDNs, as described in Section 2.3, effectively improved classification of PUAs. Figure 3 gives an example of the FQDNs and the effect of common FQDN removal. These were extracted in *probably a variant of Android/Adware.Wooboo.C*, which is one of the PUA varieties with the highest classification accuracy. At an R of 0.01, this increased the accuracy by 2.6% over the case without common FQDN removal, achieving the highest accuracy of 79.7%. However, the ACC did not always take a higher value as R was decreased. We assumed that an FQDN appearing in the list of common FQDNs was specific to some varieties of PUA. We will discuss the issue of common FQDN removal in Section 4. Note that both the training data and testing data decreased as R was decreased. In this case, from the 237 varieties of PUA, 227 were selected as testing data after the random sampling and sample reduction described in Section 2.5. The accuracy was also computed for each variety of PUA. Of the 227 varieties, 157 (69.2%) had an ACC greater than or equal to 0.7, and 10 (4.4%) had ACC values lower than 0.3. We were able to classify a sample within 4.2 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: 32GB 1333MHz UDIMM). Table 6 shows the classification accuracy for each variety of malware. The highest accuracy of 87.4% was achieved at $R = 0.02$ for 128 of the 393 varieties of malware covered by the testing data.

3. http://www.domob.cn/

4. Discussion

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

App platform: This study focused only on the Android platform, with a partial analysis of Windows. Other platforms, including iPhone and Windows Mobile, were not considered. To extend the coverage, apps will need to be collected from the 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: We selected PUA varieties based on keywords in the name that could be detected by AV. This is clearly vendor specific. AVClass [9] is a technique that allows a common name to be assigned by summarizing the names detected by multiple AVs. As the accuracy of classification depends completely on the use of appropriate labels, this technique may be used to improve the methods presented in this paper.

FQDN extraction: As described in Section 2.2, we extracted FQDNs from all possible URLs accessed by the app. However, the app will not necessarily access all these

TABLE 4. TOP 5 EXAMPLES OF PUAS MISCLASSIFIED AS MALWARE.

# of misclassification	misclassified PUA in testing data	predicted malware in training data
14	*a variant of Android/Domob.G potentially unwanted*	*probably a variant of Android/Domob.G*
12	*a variant of Android/AdMogo.A potentially unwanted*	*probably a variant of Android/Domob.G*
12	*a variant of Android/AdDisplay.AdsWo.A potentially unwanted*	*a variant of Android/Domob.F*
10	*a variant of Android/Domob.B potentially unwanted*	*probably a variant of Android/Domob.B*
9	*a variant of Android/AdDisplay.AdsWo.A potentially unwanted*	*probably a variant of Android/Domob.G*

TABLE 5. CLASSIFICATION ACCURACY BETWEEN VARIETIES OF PUA.

R (# of removal FQDN)	ACC	mean J	max. J	min. J	med. J	std J	mean # of train	mean # of test	mean time [s]
0.01 (134)	**0.797**	**0.615**	**0.963**	**0.090**	**0.600**	**0.185**	**8329.8**	**489.0**	**4.2**
0.02 (63)	0.773	0.640	0.971	0.099	0.607	0.195	8618.3	501.3	3.3
0.03 (44)	0.793	0.666	0.972	0.107	0.667	0.182	8936.8	533.6	4.9
0.04 (35)	0.787	0.672	0.971	0.113	0.671	0.183	9048.2	534.2	3.7
0.05 (31)	0.786	0.676	0.972	0.106	0.682	0.182	9121.6	535.2	5.1
0.06 (24)	0.772	0.681	0.974	0.119	0.705	0.184	9225.7	536.7	3.9
0.07 (19)	0.786	0.682	0.974	0.092	0.686	0.180	9330.0	539.5	4.8
0.08 (18)	0.785	0.685	0.971	0.107	0.695	0.180	9375.9	542.0	4.9
0.09 (16)	0.765	0.685	0.973	0.128	0.695	0.183	9346.3	540.2	3.4
0.10 (15)	0.771	0.687	0.975	0.096	0.695	0.182	9363.6	541.3	4.0
w/o removal	0.771	0.728	0.974	0.136	0.750	0.172	9994.3	542.7	4.8

TABLE 6. CLASSIFICATION ACCURACY BETWEEN VARIETIES OF MALWARE.

R (# of removal FQDN)	ACC	mean J	max. J	min. J	med. J	std J	mean # of train	mean # of test	mean time [s]
0.01 (134)	0.866	0.680	0.961	0.076	0.714	0.176	8314.3	480.0	3.5
0.02 (63)	**0.874**	**0.712**	**0.968**	**0.069**	**0.740**	**0.172**	**8618.0**	**500.3**	**3.8**
0.03 (44)	0.870	0.731	0.969	0.066	0.756	0.160	8937.2	510.3	4.5
0.04 (35)	0.863	0.740	0.971	0.078	0.770	0.160	9023.0	517.5	3.8
0.05 (31)	0.862	0.742	0.970	0.062	0.777	0.158	9087.8	520.2	4.5
0.06 (24)	0.865	0.752	0.972	0.068	0.784	0.157	9176.1	517.4	5.0
0.07 (19)	0.863	0.755	0.970	0.072	0.800	0.161	9325.1	522.0	3.5
0.08 (18)	0.857	0.753	0.970	0.097	0.799	0.157	9321.9	521.5	5.3
0.09 (16)	0.861	0.759	0.970	0.084	0.803	0.158	9374.4	518.2	3.5
0.10 (15)	0.856	0.755	0.971	0.082	0.801	0.159	9378.2	519.2	5.3
w/o removal	0.841	0.795	0.972	0.114	0.838	0.153	9986.1	518.0	5.7

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.

Common FQDN removal: The classification accuracy was evaluated across a limited range of R values, which is one parameter of document frequency. Evaluation may improve further at R values smaller than 0.01. However, since the number of apps will decrease as R decreases, a trade-off arises between the PUA coverage and classification accuracy. As discussed in Section 3.2, common FQDN removal may eliminate FQDNs that are specific to particular varieties of PUA. To classify PUAs more accurately, completely excluding PUAs from the samples for building a list of common FQDNs may be needed. Other algorithms can also be used to build an appropriate list of common FQDNs.

5. Related Studies

The attention given to PUAs has increased over recent years. Previous studies have focused on analysis and detection of Android malware, and ours is the first to measure and classify the DNS queries generated by PUAs using a large-scale dataset. Here, we review some recent studies of Android malware detection and PUAs.

Detection of Android malware: MADAM [10] 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 made between patterns and features. More than 96% of malicious apps were detected from 2,800 apps, with a low false positives rate. CREDROID [11] was designed to detect Android malware using Virus Total 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.

Taxonomy of PUAs: Zhou et al. [12] systematically characterized the same dataset used in [11], 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. [13] introduced a structured PUA taxonomy for mobile apps that defined PUA classification criteria, for use by security vendors and testing organizations.

Distribution of PUAs: Kotzias et al. [14] 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. [15] 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. [16] developed lightweight heuristics for triage of mobile grayware that leverages text analytics and static program analysis. Nine categories of grayware were defined, based on installation and runtime behavior. A large-scale study of grayware on Google Play was conducted, and the effectiveness of triage was demonstrated on reducing from1 million apps to tens of apps. Chen et al. [17] analyzed potentially harmful libraries (PhaLib) across the Android and iOS platforms. These were repackaged as legitimate libraries and propagated as malware. The approach used was to map, Android PhaLib to iOS libraries, and to correlate suspicious behavior. They identified 117 Android PhaLibs in a search of over 1.3 million Android apps, and 46 PhaLibs in 140,000 iOS apps.

6. Conclusion

In this work, we performed the first systematic measurement and classification of PUAs, based only on the use of DNS queries. We first extracted the FQDNs from the DNS queries generated by the app, and built a list of common FQDNs, based on document frequency. 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 ineffective for detection and classification. From these results, we predicted the category and variety from the training dataset for each testing data when the highest similarity value had been obtained. To reduce computational complexity, duplicated sets of FQDNs within the same variety were removed. This allowed Android PUAs, malware, and benign apps to be easily distinguished with approximately over 90% accuracy. More than 200 varieties of PUA are classified correctly with 79.7% accuracy, with a computational time of 4.2 s per app. The study provides evidence that Android malware can also be classified accurately, using the same methodology.

Acknowledgments

We thank Mr. Yuta Ishii and Mr. Sho Mizuno for collecting valuable datasets. A part of this work was supported by JSPS Grant-in-Aid for Scientific Research B, Grant Number JP16H02832

References

[1] "Threat intelligence report - Unwanted software," https://www.microsoft.com/security/portal/enterprise/threatreports_october_2015.aspx.

[2] "VirusTotal - Free Online Virus, Malware and URL Scanner," https://www.virustotal.com/.

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

[4] "Cuckoo Sandbox: Automated Malware Analysis," https://www.cuckoosandbox.org/.

[5] "Alexa Top Sites," http://www.alexa.com/topsites.

[6] "Adblock Plus," https://easylist-downloads.adblockplus.org/easylist.txt.

[7] "Blocking with ad server and tracking server hostnames," https://pgl.yoyo.org/as.

[8] "DNS-BH - Malware Domain Blacklist," http://www.malwaredomains.com/.

[9] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVClass: A Tool for Massive Malware Labeling," in *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID 2016, Sep. 2016, p. 1—22.

[10] 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. PP, no. 99, pp. 1–1, 2016.

[11] J. Malik and R. Kaushal, "CREDROID: Android Malware Detection by Network Traffic Analysis," in *Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing*, ser. PAMCO '16, Jul. 2016, pp. 28–36.

[12] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in *2012 IEEE Symposium on Security and Privacy*, ser. S&P 2012, May 2012, pp. 95–109.

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

[14] P. Kotzias, L. Bilge, and J. Caballero, "Measuring PUP Prevalence and PUP Distribution through Pay-Per-Install Services," in *Proceedings of the 25th USENIX Security Symposium*, Aug. 2016, pp. 739–756.

[15] 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," in *Proceedings of the 25th USENIX Security Symposium*, Aug. 2016, pp. 721–739.

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

[17] 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," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, ser. S&P 2016, May 2016, pp. 357–376.