POSTER: Insights of Antivirus Relationships when Detecting Android Malware: A Data Analytics Approach

Ignacio Martín, José Alberto Hernández
Universidad Carlos III de Madrid
Avda de la Universidad 30, 28911 Leganés
Madrid, Spain
{ignmarti, jahgutie}@it.uc3m.es

Sergio de los Santos, Antonio Guzmán
Telefónica Digital Identity & Privacy
Ronda de la Comunicación s/n
Madrid, Spain
{ssantos,
antonio.guzman}@11paths.com

ABSTRACT

This work performs a deep analysis on the behaviour of Anti-Virus (AV) engines regarding Android malware detection. A large dataset, with more than 80K apk files tagged as Malware by one or many AV engines is used in the analysis. With the help of association rule learning, we show interesting patterns and dependencies between different AV engines.

1. INTRODUCTION

According to the latest annual mobility report by Ericsson, there are 2.6 billion smartphone subscriptions globally, and about 75% of them use the Android operating system. Only last year, malware applications grew three times more than they did in 2014[1].

At present, malware detection has become a complex process involving a wide range of techniques, such as static and dynamic analysis, heuristics, cloud computing or anomaly detection, all aiming to create signatures and real-time automatic classification of malware [7]. Unfortunately, not a single AV has been proved to be able to reach 100% accuracy on the detection of malware. In fact, the authors in [2] show that using several AV engines rather than one yields more accurate detection results. Thus, collaboration among AVs is essential and, as a consequence, a number of online virus security services, like VirusTotal, Andrototal or Metascan have appeared allowing users to check suspicious files against the most popular AV engines.

Prior research has been carried out over multi-scanner platforms. For instance, the authors in [1] develop a comparison of AV engines from VirusTotal by checking the output with a malware dataset and model AV confidence using a hyper-exponential curve. In [3], temporal analysis on VirusTotal AV labels and their evolution (when a sample is labeled or the label is withdrawn) is performed using a collection of malware applications obtained through a honeypot network. Other authors, such as the ones in [4] propose different methodologies to improve malware labelling and check their applicability to the temporal evolution of labels from VirusTotal using a large application dataset . Furthermore, some works such as [6, 5] address the different problems arising for categorizing and classifying malware, such as lack of naming standards or even lack of consensus.

This work analyses a large and wide dataset of Android malware applications, namely more than 80K applications tagged by at least one of the 61 most popular AV engines (i.e. McAffee, Trend Micro or Karspersky, etc), with the goal to identify patterns in how such AV engines correlate and behave.

2. DATASET DESCRIPTION AND ANALYSIS

The dataset under study comprises exactly 82,866 labelled applications collected from Google Play by the TACYT application[2]. We shall use matrix A (with dimensions 82,866 x 61), whose elements $A_{ij} \in \{0, 1\}$ are labels denoting whether or not the i-th application has been tagged as Malware by the j-th AV engine. The AV companies have been anonymized for privacy reasons. In what follows, the AV engines will be referred to as AV1, AV2,..., AV61.

According to the dataset, many AV engines often agree on their decisions; however, in most cases a given apk is flagged just by a single AV engine. Actually, the distribution of detections per application presents a Zipf-like pattern, where the number of single-detection applications is 38,933, the mean number of detections is 3,125, the median is 2 and the maximum count of detections registered for an application is 53. Fig. 3 shows the percentage of malware applications covered by combining different AV engines. As shown, more than 40% applications are covered by the most active engine (i.e. AV27). The second AV such that its union with AV27 covers most malware apps is AV2, reaching 68.59% of the total. Next one is AV58 such that AV27 $\cup$ AV2 $\cup$ AV58 reach 74.79%, and so on. As shown, the union of 12 AV engines covers around 95% of the applications, while the union of 24 covers more than 99% of the total malware dataset.

[1]See: http://www.kaspersky.com/about/news/virus/2016/The_Volume_of_New_Mobile_Malware_Tripled_in_2015

[2]https://www.elevenpaths.com/es/tecnologia/tacyt/index.html, last access May 2016.

CCS'16 October 24-28, 2016, Vienna, Austria

ACM ISBN 978-1-4503-4139-4/16/10.

DOI: http://dx.doi.org/10.1145/2976749.2989038

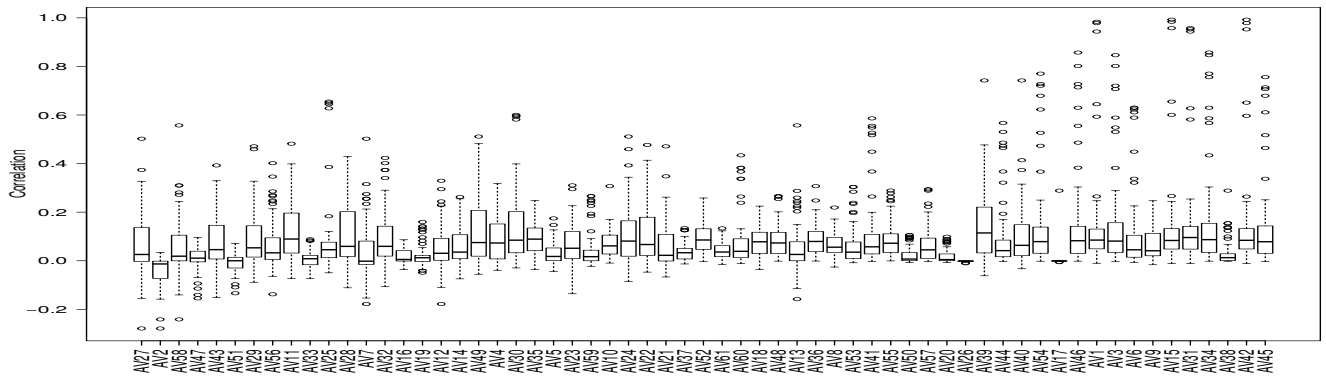

Figure 1: Correlation boxplots between AV engines

2.1 Correlation between engines

Fig. 1 shows the box-plots of the correlations observed among engines. The engines are sorted by joint detection count, that is, the AV which together with the previous ones provides highest coverage. As shown, most correlation values observed between AV engines are weak (below 0.3) or, in some cases, moderate (0.3 to 0.5). For example, the most popular AV engine (AV27) shows correlation values between -0.2 and 0.5 with other AV engines, however most values concentrate between 0 and 0.2. In spite of these correlation results, we identify six AV engines that show almost no correlation with any other one, these are: AV2, AV47, AV51, AV33, AV16 and AV19. We shall refer to them as "eccentric" AV engines. Additionally, we observe that the last AV engines show strong correlations with some others (values close to one). Essentially, from AV46 onwards, these engines show correlation values above 0.8 with some other engines. We shall call AV46, AV1, AV3, AV6, AV9, AV15, AV31, AV34, AV38, AV42 and AV45 as "redundant" AV engines.

No	lhs	rhs	support	conf.
1	AV58	AV27	0.2134	0.620
2	AV27	AV58	0.2134	0.520
3	AV7	AV27	0.2036	0.850
4	AV27	AV7	0.2036	0.496
5	AV13	AV58	0.1420	0.990
6	AV58	AV13	0.1420	0.412
7	AV32	AV27	0.1235	0.858
8	AV27	AV32	0.1235	0.301
9	AV43	AV27	0.1181	0.660
10	AV27	AV43	0.1181	0.288
11	AV7	AV58	0.1173	0.490
12	AV58	AV7	0.1173	0.341
13	AV56	AV27	0.1124	0.712
14	AV27	AV56	0.1124	0.274
15	AV28	AV27	0.1087	0.784
16	AV27	AV28	0.1087	0.265
17	AV58,AV7	AV27	0.1066	0.908
18	AV27,AV7	AV58	0.1066	0.523
19	AV27,AV58	AV7	0.1066	0.499
20	AV13	AV27	0.1000	0.698

Table 1: 20 most relevant association rules.

2.2 Association Rule Learning

Association rule learning is a very popular data mining method for discovering interesting relationships among variables in large databases. In particular, this method has been extensively used in market basket analysis to analyse patterns in supermarket purchase behaviours of clients. Rule mining or association mining seeks for item-sets maximising the following metrics:

- Support: is computed as the frequency of an item with respect to the total.

$$Support(X) = \frac{\text{Frequency of item X}}{\text{Total number of transactions}}$$

- Confidence: shows how frequently a certain rule appears within all detections with respect to the observed behavior.

$$Conf.(X \rightarrow Y) = \frac{Support(X \cap Y)}{Support(X)}$$

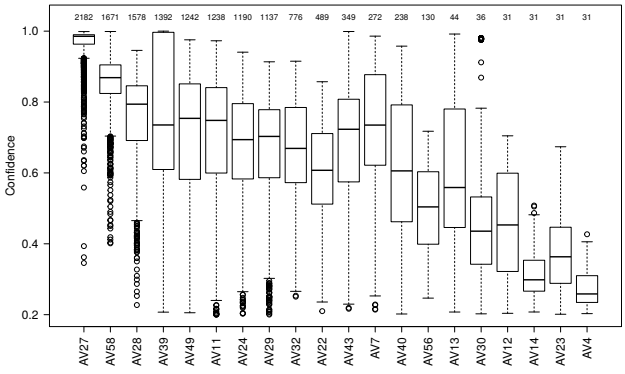

Figure 2: Boxplot of confidence measures of the most relevant followers (total number of appearances on top)

Table 1 shows the 20 most relevant association rules found in our dataset, sorted by support. For instance, rule no. 1 $\{AV58\} \rightarrow \{AV27\}$ reveals that, when AV58 detects a malware application (which occurs in 21.3% of the cases), then 62% of the times AV27 tags the same application as malware too (conf. = 0.62). The opposite occurs in 52% of the cases

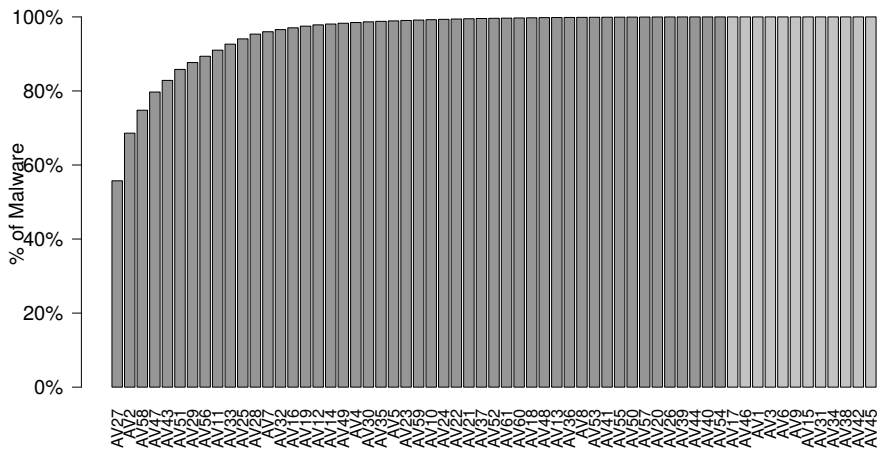

Figure 3: Sorted AV engines by their joint detection rate

(this is rule no. 2). We also observe that some AV engines appear several times in the table, participating in many association rules. For instance, AV27 appears with AV58, but also with AV7 (rule no. 3), AV32 (rule no. 5) and AV43 (rule no. 7). Other interesting patterns identified are: AV58 appears with AV28, AV56 and AV32, etc. Moreover, we have selected those association rules with a minimum support of 1% and a minimum confidence of 20%. There are exactly 20,198 rules meeting this criteria. Fig. 2 shows box-plots of confidence values for the AV engines found on the right-hand side (rhs) of these rules. There are 20 AV engines which have very strong association with others, namely: AV27, AV28, AV39, AV11, AV24, AV29, AV32, AV22 and AV40. Besides, it is worth noticing that our previously defined eccentric engines do not appear in these inferred rules. This is a rather obvious results since such eccentric engines do not correlate with any other.

3. CONCLUSIONS AND FUTURE WORK

In conclusion, this comparative analysis of AV engines on a large Android malware dataset has shown that: (1) Most AV engines are weakly correlated with the others; however some AV engines show moderate to high correlation. (2) There is a particular group of AV engines that behave "eccentrically", that is, not correlated with any other engine in the dataset. (3) Association rules revealed that certain malware engines very often tag malware applications together, showing a strong dependency or "following" behaviour. Future work will attempt to define a mechanism to combine AV engines weighted by confidence. Essentially, the goal is to identify a metric of "malwarish" confidence, given the observed fact that some AV engines are highly correlated ("followers") while others do the opposite as the majority ("eccentric" ones).

4. ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the project TIGRE5-CM project (grant no. S2013/ICE-2919) funded by the "Comunidad Autónoma de Madrid" Government plan "R&D programs among research Groups", the national project BigDatAAM (FIS2013-47532-C3-3-P), funded by the Ministerio de Economía y Competitividad of SPAIN through, and the EU-funded H2020 TYPES project (grant no. H2020-653449).

5. REFERENCES

[1] P. Bishop, R. Bloomfield, I. Gashi, and V. Stankovic. Diversity for security: A study with off-the-shelf antivirus engines. In *IEEE 22nd Int. Symp. on Software Reliability Engineering*, 2011.

[2] M. Cukier, I. Gashi, B. Sobesto, and V. Stankovic. Does malware detection improve with diverse antivirus products? an empirical study. In *32nd Int. Conf. on Computer Safety, Reliability and Security. IEEE*, 2013.

[3] I. Gashi, B. Sobesto, S. Mason, V. Stankovic, and M. Cukier. A study of the relationship between antivirus regressions and label changes. In *IEEE 24th Int. Symp. on Software Reliability Engineering*, 2013.

[4] A. Kantchelian, M. C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, A. D. Joseph, and J. Tygar. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proceedings of the 8th ACM Workshop on AI and Security*, 2015.

[5] Y. Le Traon. On the lack of consensus in anti-virus decisions: Metrics and insights on building ground truths of android malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment: DIMVA 2016, San Sebastián, Spain, 2016, Proceedings*, volume 9721, page 142. Springer, 2016.

[6] F. Maggi, A. Bellini, G. Salvaneschi, and S. Zanero. Finding non-trivial malware naming inconsistencies. In *Int. Conf. on Information Systems Security*. Springer, 2011.

[7] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur. Survey on malware detection methods. In *Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK'09)*, 2009.