Spyware Prevention by Classifying End User License Agreements

Niklas Lavesson, Paul Davidsson, Martin Boldt, and Andreas Jacobsson

Blekinge Institute of Technology,
Dept. of Software and Systems Engineering,
Box 520, SE–372 25 Ronneby, Sweden
{niklas.lavesson,paul.davidsson,
martin.boldt,andreas.jacobsson}@bth.se
http://www.bth.se/tek/disl

Abstract. We investigate the hypothesis that it is possible to detect from the End User License Agreement (EULA) if the associated software hosts spyware. We apply 15 learning algorithms on a data set consisting of 100 applications with classified EULAs. The results show that 13 algorithms are significantly more accurate than random guessing. Thus, we conclude that the hypothesis can be accepted. Based on the results, we present a novel tool that can be used to prevent spyware by automatically halting application installers and classifying the EULA, giving users the opportunity to make an informed choice about whether to continue with the installation. We discuss positive and negative aspects of this prevention approach and suggest a method for evaluating candidate algorithms for a future implementation.

Key words: EULA spyware classification prevention

1 Introduction

The amount of spyware has increased substantially over the last years, much due to the mainstream usage of Internet. Spyware is designed to collect user information for marketing campaigns without the informed consent of the users. This type of software is commonly secretly included with popular applications available for public download and it is typically difficult to remove once it has infected a computer system. Consequently, we recognize the need for an efficient prevention method. Spyware distributors usually disguise their software as legitimate to reach as many users as possible. However, to avoid legal repercussions they often mention in the End User License Agreement (EULA) that spyware will indeed be installed. However, this information is given in a way most users find hard to understand. Even legitimate EULAs can be hard to fully comprehend due to their length and the juridical terminology used.

We investigate whether it is possible to take advantage of the fact that the installation of spyware is mentioned in the EULAs and address the spyware problem by applying supervised learning algorithms to classify EULAs of both

legitimate and spyware-hosting applications in order to determine if it is possible to detect from the EULA whether the associated software hosts spyware or not.

1.1 Related Work

Adaware is an anti-spyware product [2] that can be used to scan computers for spyware and remove the found instances. However, whereas most anti-spyware products are reactive, thus trying to remove something which has already infected a system, we examine the possibility of a preventive approach that detects the presence of spyware by examining the EULA, even before the application is installed. There are a few applications available for this purpose today. We have identified one web-based tool called the EULA analyzer [3]. The user of the tool pastes the content of a EULA into a text field and the tool then analyzes the EULA and assigns a credit point that could indicate the likelihood of spyware inclusion in a particular application. It also shows some statistics about language complexity. However, it is important to notice that the credit point merely corresponds to the number of spyware related keywords or phrases that are found. There is no automatic learning of new patterns, there is no way to represent, e.g., non-trivial rules. Perhaps most notably, the tool makes no classification. Thus, this crucial task is left entirely to the user. A low credit point supposedly indicates legitimate software and a high point indicates spyware. The credit limit that separates bad EULAs from good EULAs is dependent on the particular set of submitted EULAs and has to be manually found by the user in order to be able to use the EULA analyzer for classifying software applications based on their corresponding EULAs.

Another product, with similar functionality, is EULAlyzer [4]. EULAlyzer is an installable application and the main difference in functionality, in comparison to EULA Analyzer, is only available in the professional version of the software. This functionality, called EULA watch, halts user-oriented application installers and analyzes the application EULA automatically. Unfortunately, both state-of-the-art tools are proprietary and thus it is impossible to know for sure which techniques or algorithms are used. However, the tool descriptions indicate that they rely on a static set of keywords. We have not been able to identify any studies that apply supervised learning algorithms to classify software on the basis of their EULAs. However, much work has been done in the related area of E-mail filtering, i.e., the classification of E-mail messages as legitimate or spam depending on the subject, body or other properties, using different learning algorithms, e.g.: rule learners and support vector machines [5, 6]. A more recent study investigates the performance of random forests for the same type of problem, claiming that this algorithm outperforms some of the earlier mentioned algorithms on several problems [7]. We have previously outlined a large number of related studies, which have been conducted within the security research field [1].

1.2 Outline

The remainder of our study is organized as follows. In Sect. 2 we describe the data gathering process and the experiment. This is concluded by a review of the experimental results. Section 3 contains a discussion about some consequences of these results. Finally, conclusions and pointers to future work are reviewed in the last section.

2 Experiments

Our hypothesis is that it is possible to detect from the EULA whether the associated software contains spyware or not, i.e., if it should be classified as good or bad. It is obvious that one could employ a nominal scale of more than two alternatives, for instance to indicate different threat levels. However, we argue that two alternatives are enough to investigate the stated hypothesis.

2.1 Data Collection

Our classification problem is quite analogous to that of classifying E-mail messages as either spam or legitimate. Thus, we choose to adopt a simple, yet successful way of representing the data from this area of research by representing each EULA with a word frequency vector. Thus, the data instances are represented by pairs of word frequency vectors and classes. The following search strategy was adopted to collect applications; they should be easily downloaded from the Internet and present the user with a EULA that could be copied and pasted as ASCII text. The good software instances were collected by downloading the 50 most popular Windows applications from Download.com [8] and the bad applications were collected from SpywareGuide.com [9]. After the installation of each application the operating system was scanned with Adaware to verify the classification [2]. This verification showed that all applications associated with bad EULAs were detected by Adware, while no hits were found for the legitimate applications.

2.2 Data Representation

We stored the data set using the Weka ARFF format in which each word frequency is represented by a numeric attribute and the class is represented by one nominal attribute with two possible values; good or bad [10]. The data set features 50 instances classified as good and 50 instances classified as bad, thus we did not have to deal with problems associated with a skewed class distribution. However, we believe that an equal class distribution will be difficult to achieve when creating larger data sets due to the simple fact that it is much harder to find bad applications and their corresponding EULAs. A future study could instead try to achieve a distribution that is close to the real-world distribution, which is generally perceived to have a much higher ratio of good software. The

word frequency vector was generated using Wekas StringToWordVector filter with the settings adjusted as in the study by Frank and Bouckaert [11]. Thus, the TF IDF weight was applied, all characters were converted to lowercase, only alphabetic tokens were considered, and stop words as well as hapax legomena were removed. We furthermore employed a form of feature selection in that we used the Weka default setting of storing a maximum of 1000 words per class to generate the data set.

2.3 Algorithm Evaluation

The main objective of this paper is not to determine the most suitable algorithm for the studied problem, which would most certainly involve extensive parameter tuning of each featured algorithm, but rather we want to determine if the formulated hypothesis should be accepted or not. To maximize the probability of finding a useful pattern we included a diverse population of 15 algorithms from different learning categories (e.g. perceptron and kernel functions, lazy learners, Bayesian learners, trees, meta-learners, rules, etc.). We used Weka algorithm implementations and applied the default configuration for each algorithm. To test the hypothesis we needed to assess the accuracy of the algorithms.

Since we had a limited amount of data for training and testing (100 instances), we chose to perform repeated holdout tests to estimate prediction accuracy using two metrics; accuracy (correctly classified instances divided by total number of classified instances), and the area under the ROC curve (AUC). These metrics are by far the most widely used although one should keep in mind that there are issues both regarding accuracy and AUC as with most other metrics [12]. Many studies have shown the applicability of AUC for a wide range of learning problems, cf. Provost and Fawcett [13]. Intuitively, if our hypothesis holds, it should be possible to generate a classifier that performs better on average than randomly guessing the class. Hence, we formulate the hypothesis test as follows; if anyone of the featured algorithms is significantly better than a random guesser on the featured data set for both accuracy and AUC we accept the hypothesis, otherwise we reject it.

To investigate which algorithms performed significantly better than a random guesser we used repeated holdouts and the corrected paired t-test, which is a common setup used in similar applications [14]. We calculated the mean and standard deviation of 10 repeated holdouts with a 66% training set / 34% test set randomized split for each of the following metrics; accuracy (percent correct), AUC (including true positives rate and false positives rate), training time, and testing time. We used the corrected paired t-test (confidence 0.05, two-tailed) to compare each featured algorithm with a Weka baseline classifier called ZeroR, which classifies all instances as belonging to the same class. Thus, it shares the same results for both accuracy and AUC with a random guesser for a Boolean problem.

2.4 Comparison with the State-of-the-art

We also compared the performance, in terms of accuracy, of the 15 featured algorithms with the state-of-the-art EULA analyzer tool according to the following procedure; we generated ten folds for testing by sampling, without replacement, 17 bad instances and 17 good instances for each fold (since the holdout procedure used to evaluate the 15 algorithms uses a 66/34 split) from the collection of EULAs. Obviously we did not generate any training folds since the EULA analyzer is a static model in the sense that it does not learn. Since the EULA analyzer works by looking for keywords in plain text we used text document instances instead of word vector representations, which have been subjected to, e.g., feature selection. Thus, it is important to keep this difference in mind when comparing the results later. More importantly, one should recognize that in a real-world scenario, the accuracy of EULA analyzer is dependent on the interpretation capabilities of the user concerning the resulting credit score for a particular EULA. For our experiment we used the optimal credit score cut-point which means that the published accuracy results of the EULA analyzer are more than likely to be higher than what could be achieved by the average user.

Since the testing folds for the EULA analyzer and the testing folds for the algorithms are not identical, we used a corrected non-paired t-test (confidence 0.05, two-tailed) for this part of the experiment. The objective was to find out for which algorithms there are significant improvements or degradations in performance compared to the state-of-the-art. However, as clearly mentioned earlier, we did not try to tune any of the learning algorithms to maximize performance (i.e., increasing the probability of finding significant improvements over the state-of-the-art). A secondary priority was to measure performance in terms of training and testing time. These priorities coincide with the objective of investigating if indeed a tool can be designed that builds upon the classification method presented in this paper to prevent the covert installation of spyware.

2.5 Experimental Results

The results are presented in Tab. 1. It is clear that our hypothesis should be accepted since at least one classifier achieves a significant improvement, with regard to both AUC and accuracy, in comparison to the baseline classifier. There are, in fact, significant improvements of both accuracy and AUC for 13 out of 15 featured algorithms (Ridor and DecisionStump are the only exceptions). Moreover, when comparing with the accuracy of the state-of-the-art method it is also shown that 10 algorithms outperform this method on the studied data set. However, only the improvements of NaiveBayesNominal (abbrievated NaiveBayesNom in Tab. 1) and SMO (Support Vector Machines) are statistically significant. For one algorithm, KStar, the accuracy is significantly degraded in comparison to the state-of-the-art. The high false positive rate of Kstar (0.77) might be alarming but it is important to recognize that our study merely features 100 instances. As more data is gathered for future work our guess is that the performance will increase even for the worst performing algorithms.

Table 1. Experimental results in terms of: accuracy, true positives rate (TPR), false positives rate (FPR), area under the ROC curve (AUC), training time, testing time, and CEF. Significant improvements in accuracy, compared to the EULA analyzer, are shown with + while significant degradations are shown with −.

Algorithm	Accuracy	TPR	FPR	AUC	Training Time	Testing Time	CEF
AdaBoostM1	0.74(0.06)	0.72(0.08)	0.24(0.09)	0.78(0.04)	3.55(0.28)	0.00(0.01)	0.67
DecisionStump	0.69(0.11)	0.54(0.16)	0.16(0.19)	0.69(0.11)	0.33(0.08)	0.00(0.00)	0.00
HyperPipes	0.76(0.08)	0.91(0.14)	0.38(0.19)	0.90(0.07)	0.04(0.01)	0.07(0.09)	0.00
IBk	0.78(0.06)	0.71(0.07)	0.15(0.10)	0.78(0.06)	0.04(0.01)	0.13(0.02)	0.72
J48	0.73(0.10)	0.72(0.16)	0.26(0.18)	0.73(0.10)	1.29(0.23)	0.00(0.01)	0.81
JRip	0.71(0.05)	0.71(0.14)	0.29(0.12)	0.72(0.07)	2.02(0.23)	0.00(0.00)	0.79
KStar^{-}	0.60(0.04)	0.96(0.03)	0.77(0.09)	0.68(0.07)	0.00(0.00)	9.20(0.42)	0.00
NaiveBayes	0.79(0.10)	0.91(0.09)	0.32(0.17)	0.80(0.10)	0.31(0.02)	0.11(0.05)	0.00
NaiveBayesNom^{+}	0.88(0.06)	0.88(0.11)	0.12(0.11)	0.93(0.06)	0.03(0.01)	0.00(0.01)	0.77
PART	0.73(0.11)	0.72(0.15)	0.26(0.15)	0.72(0.11)	2.41(2.15)	0.00(0.01)	0.81
RandomForest	0.75(0.07)	0.79(0.09)	0.28(0.09)	0.83(0.08)	3.64(0.20)	0.00(0.00)	0.68
RBFNetwork	0.77(0.08)	0.75(0.12)	0.21(0.13)	0.78(0.09)	1.46(0.19)	0.17(0.02)	0.71
Ridor	0.68(0.11)	0.63(0.14)	0.28(0.13)	0.68(0.11)	0.87(0.11)	0.00(0.01)	0.00
SMO^{+}	0.84(0.04)	0.78(0.08)	0.11(0.08)	0.84(0.04)	0.25(0.08)	0.00(0.00)	0.75
VotedPerceptron	0.81(0.07)	0.85(0.13)	0.22(0.10)	0.87(0.07)	0.04(0.01)	0.02(0.01)	0.72
ZeroR (baseline)	0.50(0.00)	1.00(0.00)	1.00(0.00)	0.50(0.00)	0.00(0.01)	0.00(0.00)	0.00
EULA analyzer	0.73(0.04)	0.71(0.05)	0.22(0.12)	0.80(0.08)	N/A	N/A	N/A

We further observe that NaiveBayesNominal is the best performing algorithm, achieving the best AUC and accuracy followed by SMO and VotedPerceptron. SMO has the highest training time of these three candidates; however, the testing time does not differ significantly between them. Only KStar stands out, with regards to the measured testing time, with a mean result of approximately 10 seconds, while the other algorithms achieve results close to zero seconds. Regarding training time, there is no algorithm needing more than 5 seconds and, in particular, HyperPipes, IBk, KStar, NaiveBayesNominal, and VotedPerceptron need close to zero seconds. Exactly what words the best behaving algorithms used for distinguishing between good and bad software EULAs could not be easily grasped because of their implicit representation of the learned classifiers. However, in our limited data set of 100 instances, tree and rule based algorithms identified combinations of words such as 'search' and 'advertisements' for distinguishing between good and bad EULAs.

2.6 Multi-criteria Evaluation

One of the most fundamental properties of an intended prevention tool would arguably be to minimize the possibility of misclassifying spyware-hosting applications as legitimate, since those misclassifications could result in an increased security risk. Consequently, we prefer as candidates for implementing this tool, those algorithms that achieve a low FPR. Overall, algorithms included in the prevention tool should be as accurate as possible. In order to ensure high accuracy and a low FPR we consider the use of multi-criteria optimization and evaluation of learning algorithms.

For this purpose we suggest the generic multi-criteria metric, CEF [18], which can be used to trade-off multiple evaluation metrics when evaluating or selecting between different learning algorithms or classifiers. Each included metric can be associated with an explicit weight and an acceptable range. When analyzing the prevention tool requirements it is clear that we need to use evaluation metrics for accuracy (of classifying both good and bad applications), time (classification response time), and explainability (for visualization). Mapping the requirements to the available experiment data and making informed choices (see [18]) about bounds and explicit weighting, we can calculate the CEF score for all algorithms included in the experiment. This score is presented in the last column of Tab. 1. For the selected setting, 6 algorithms get a CEF score of 0 since at least one metric value for each algorithm was outside of the acceptable range. JRip and PART are ranked higher than NaiveBayesNominal since they scored higher in explainability.

3 Discussion

The experiment raises several interesting issues which will now be addressed. We first bring forth some technical aspects related to the featured algorithms and their performance on EULA classification, and continue discussing the importance of automatic EULA analysis. This is followed by a proposal of a novel software tool for spyware prevention. The results of the top three candidates seem to be well-aligned with results in related work; NaiveBayesNominal is known to perform very well on large vocabularies [15]. However, it is usually acknowledged that SMO outperforms NaiveBayesNominal on many problems [16]. Still, it should be considered that NaiveBayesNominal does not need to be tuned for a particular problem, while SMO includes a large number of configurable parameters. This would favor the former algorithm in this study since only default configurations are used.

It is typically hard for the average user to know if an application hosts spyware or not. It is evident that many users would benefit from using an automated tool, which could assist them in analyzing the EULA by predicting if the related software hosts spyware or not. In order to implement a successful tool, it is neccessary to collect data for a larger empirical experiment since a data set consisting of 100 instances is only acceptable for an initial study. It is also plausible to assume that legitimate applications greatly outnumber the applications that host spyware in a real-world setting. Thus, it would be beneficial to perform the larger experiment using a data set with a skewed class distribution in order to properly represent this assumed setting. In other words, if 1000 EULAs are collected, it may be sufficient if only 5% or 10% of them are spyware EULAs.

3.1 A Novel Tool for Spyware Prevention

The EULA classification method outlined in this paper can be implemented as a software tool for spyware prevention. We argue that this tool should be designed

as a middleware that operates between the operating system and the application installers. The tool should be executed as a background process set to identify and analyze a EULA as soon as it appears on the screen during an installation. Based on the result from the EULA analysis, the tool will provide the user with recommendations about the classification of the application. This allows the tool to assist users in making informed decisions about the installation of software without forcing them to read (and understand) the lengthy and intricate EULAs. Should a EULA be classified as bad, a user can take appropriate actions against it, e.g., by disagreeing with the EULA and exiting the installation process. It should be noted that any tool based on our method should not be used in isolation, but in combination with other approaches, e.g., anti-spyware software.

One possible difference between the proposed tool and the state-of-the-art lie in the application of learning algorithms, which can be trained on large amounts of recent EULAs in order to make more accurate classifications than static keyword spotting services. Additionally, many of these learning algorithms generate classifiers which can be used for visualizing which parts of the EULA that significantly contributed to its classification as either good or bad software. This visualization could, for instance, be implemented by using extracted rules or generated trees. We outline a design of the prevention tool as follows. In order for it to work properly, there are certain requirements that need to be fulfilled. First, we need to make sure that the tool is accurate in its classifications since this is the main functionality. The tool should essentially be able to detect all, or a very high quantity of, bad software but it is also desirable that it manages to classify good software correctly. Furthermore, we want the tool to respond quickly when an application presents a EULA. However, the actual training phase could be done offline and the resulting classifier could then be downloaded by the tool periodically. Thus, there are no specific requirements related to training time. Finally, it is desirable that the tool can visualize what parts of the EULA text that prompted the actual classification. However, in relation to the earlier stated requirements, this is a secondary priority.

3.2 Arguments Against the Tool

It could be argued that, if the prevention tool was made available, the spyware authors could tailor their EULA around it. For instance, virus writers have difficulties writing viruses that avoid detection of scanners because viruses must contain executable code that cannot be arbitrarily changed. Since an EULA does not contain any executable code the spyware authors could be more creative in changing its content. This argument, however, does not hold, since the spyware authors are very aware of the fact that they need to mention in the EULA that spyware will be installed, in order to avoid legal repercussions. We simply exploit this fact and use it against the spyware distributors. Another argument could be that the intended tool would be no better than, for example, Adaware, in detecting spyware since the training set classifications are validated using this product. However, we argue that this validation is sufficient for the scope of our experiment and, more importantly, the real classification of the training

set was carried out by downloading good software from a well-known source of non-spyware hosting software and bad software from a site which only features spyware-hosting applications.

Additionally, the intention is not for the tool to be a replacement for products like Adaware but rather it would be a complement that could be used to detect spyware from EULAs in software that has not yet been classified by such products. The high false positives rate is a serious concern since the user has to be able to trust the preventative tool to be able to make the correct choice of either continuing with the installation of a particular application or not. As mentioned before, we strongly believe that future experiments with larger data sets will result in lower false positives rates for most algorithms. Nevertheless, this issue should be further addressed by optimizing potential prevention tool algorithms to minimize the false positives rate. Additionally, one could more deeply investigate the performance of ensemble learners on this problem, since they have proven to be both accurate and robust in similar settings.

4 Conclusions and Future Work

The results of the conducted experiment reveal that 13 out of the 15 featured algorithms significantly outperformed a random guesser (i.e., a user that has not read or understood the EULA). Two algorithms also significantly outperformed the state-of-the-art EULA analysis method in terms of accuracy. Most notably, NaiveBayesNominal, SMO, and VotedPerceptron achieve the highest AUC and accuracy and also share a low false positives rate, which is crucial if EULA classification should indeed be used in a preventive tool. The results strongly support our hypothesis that EULAs can indeed be used to classify the corresponding software as good or bad. Based on this conclusion and the low training and testing times of most algorithms, we also conclude that it would be quite possible to use the EULA classification method in a spyware prevention tool that classifies the EULA when it is shown to a user during an application installation. The result from such an analysis gives the user a recommendation about the legitimacy of the application before the installation continues as well as some type of visualization of what information in the EULA that triggered this classification. We also suggest a multi-criteria approach for evaluating candidate algorithms to be included into the implementation of the prevention tool and calculate test scores for demonstration purposes.

For future work we intend to implement and evaluate the proposed tool using a refined multi-criteria evaluation approach. Additionally, we will collect data for a larger experiment to further validate our hypothesis and the results in general. It would also be interesting to tune the parameters of the algorithms to find the best candidate problem solvers, cf. Lavesson and Davidsson [17]. We would also involve more algorithms, computational linguistic methods, other EULA text document representations except for word frequency vectors, as well as an analysis and visualization of words that trigger classification into legitimate software and spyware, respectively.

References

1. Boldt, M., Jacobsson, A., Lavesson, N., Davidsson, P.: Automated Spyware Detection Using End User License Agreements. In: 2nd International Conference on Information Security and Assurance. IEEE Press, New York (2008)
2. Adaware, `http://www.lavasoft.com`
3. EULA Analyzer, `http://www.spywareguide.com/analyze`
4. EULAlyzer, `http://www.javacoolsoftware.com/eulalyzerpro.html`
5. Cohen, W.: Learning Rules that Classify E-Mail. Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996)
6. Drucker, H., Wu, D., Vapnik, V.: Support Vector Machines for Spam Categorization. IEEE Transactions on Neural Networks. 10(5), 1048–1054 (1999)
7. Koprinska, I., Poon, J., Clark, J., Chan, J.: Learning to Classify E-Mail. Information Sciences. 177, 2167–2187 (2007)
8. CNET Download.com, `http://www.download.com`
9. Spyware Guide, `http://www.spywareguide.com`
10. Witten, I. H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques (2nd edition). Morgan Kaufmann, San Francisco (2005)
11. Frank, E., Bouckaert, R. R.: Naive Bayes for Text Classification with Unbalanced Classes. In: 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 503–510, Springer, Berlin/Heidelberg (2006)
12. Provost, F., Fawcett, T., Kohavi, R.: The Case against Accuracy Estimation for Comparing Induction Algorithms. In: 15th International Conference on Machine Learning, pp. 445–453, Morgan Kaufmann, San Francisco (1998)
13. Provost, F., Fawcett, T.: Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In: 3rd International Conference on Knowledge Discovery and Data Mining, pp. 43–48, AAAI Press, Menlo Park (1997)
14. Nadeau, C., Bengio, Y.: Inference for the Generalization Error. Machine Learning. 52(3), 239-281 (2003)
15. McCallum, A., Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. In: AAAI98 Workshop on Learning for Text Categorization, pp. 41–48, AAAI Press, Menlo Park (1998)
16. Kibriya, A. M., Frank, E., Pfahringer, B., Holmes, G.: Multinomial Naive Bayes for Text Categorization Revisited. In: 7th Australian Joint Conference on Artificial Intelligence, pp. 488–499, Springer, Berlin/Heidelberg (2004)
17. Lavesson, N., Davidsson, P.: Quantifying the Impact of Learning Algorithm Parameter Tuning. In: 21st National Conference on Artificial Intelligence, pp. 395–400, AAAI Press, Menlo Park (2006)
18. Lavesson, N., Davidsson, P.: Generic Methods for Multi-criteria Evaluation. In: SIAM International Conference on Data Mining (2008)