Copyright © IEEE.
Citation for the published paper:

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of BTH's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by sending a blank email message to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Detection of Spyware by Mining Executable Files

Raja Khurram Shazhad

*School of Computing, Blekinge
Institute of Technology
Box 520, SE-372 25 Ronneby,
Sweden*

rks@bth.se

Syed Imran Haider

*Custom Software Development-South,
Capgemini
Campus Gräsvik 2, 371 75 Karlskrona,
Sweden*

imran.s.haider@capgemini.com

Niklas Lavesson

*School of Computing, Blekinge
Institute of Technology
Box 520, SE-372 25 Ronneby,
Sweden*

niklas.lavesson@bth.se

Abstract—Spyware represents a serious threat to confidentiality since it may result in loss of control over private data for computer users. This type of software might collect the data and send it to a third party without informed user consent. Traditionally two approaches have been presented for the purpose of spyware detection: Signature-based Detection and Heuristic-based Detection. These approaches perform well against known Spyware but have not been proven to be successful at detecting new spyware. This paper presents a Spyware detection approach by using Data Mining (DM) technologies. Our approach is inspired by DM-based malicious code detectors, which are known to work well for detecting viruses and similar software. However, this type of detector has not been investigated in terms of how well it is able to detect spyware. We extract binary features, called *n*-grams, from both spyware and legitimate software and apply five different supervised learning algorithms to train classifiers that are able to classify unknown binaries by analyzing extracted *n*-grams. The experimental results suggest that our method is successful even when the training data is scarce.

Keywords—Spyware Detection, Data Mining, Malicious Code, Feature Extraction

I. INTRODUCTION

Programs that have the potential to violate the privacy and security of a system can be labeled as Privacy Invasive Software [1]. These programs include: spyware, adware, trojans, greyware and backdoors. They may compromise confidentiality, integrity, and availability of the system and may obtain sensitive information without informed user consent [2,3]. This information is valuable for marketing companies and also generates income for advertisers from online ads distribution through adware. This factor works as a catalyst for elevating the spyware industry [1]. Traditionally, advertisements for computer users are spread by sending spam messages but such advertisements are not targeted toward a specific segment of users as no information about the users is available to the spammers. On the other hand, data collected by spyware may be used for customized ads spread through adware to an individual user.

Originally, viruses represented the only major malicious threats to computer users and since then much research has been carried out in order to successfully detect and remove viruses from computer systems. However, a more recent type of malicious threat is represented by spyware and this threat has not been extensively studied. According to the Department of Computer Science and Engineering at the University of Washington, spyware is defined as “software that gathers information about use of a computer, usually without the knowledge of the owner of the computer, and

relays the information across the Internet to a third party location” [4]. Another definition of spyware is given as “Any software that monitors user behavior, or gathers information about the user without adequate notice, consent, or control from the user” [1]. The major difference between the definitions involves user consent, which we regard as an important concept when it comes to understanding the difference between spyware and other malicious software.

Unlike viruses, which are always unwanted, spyware can sometimes be installed with the user’s expressed consent, since it may provide some useful functionality either on its own or by an accompanying software application. Due to this reason spyware overlaps the boundaries of what is considered legal and illegal software and thus falls in a greyzone. However, in most cases, the spyware vendors do not seem to provide the user with any realistic opportunity to give an informed consent or to reject the installation of a software application in order to prevent spyware. Vendors embed spyware in regular software, which is installed with the application or by using hacking methods [5]. The installed spyware may be capable of capturing keystrokes, taking screenshots, saving authentication credentials, storing personal email addresses and web form data, and thus may obtain behavioral and personal information about users. It may also communicate system configuration including hardware and software, system accounts, location information, and information about other aspects of the system to a third party. This can lead to financial loss, as in identity theft and credit card fraud [6]. The symptoms of spyware infection vary but spyware may, e.g., show characteristics like nonstop appearances of advertisement pop-ups, open a website or force the user to open a website which has not been visited before, install browser toolbars without seeking acceptance from the user, change search results, make unexpected changes in the browser, display error messages, and so forth. Furthermore, other indications of spyware may include a noticeable change in computer performance after installation of new software, auto-opening of some piece of software or the default home page in a web-browser, a changed behavior of already installed software, the occurrence of network traffic without any request from the user, and increased disk utilization even in perceivably idle conditions [5]. Some researchers have predicted that advanced spyware can possibly take control of complete systems in the near future [7].

The awareness about spyware and its removal is considered low and outside the competence of normal users [8,9]. Even if users have anti-virus software installed, it may not be helpful

against spyware until it is designed particularly for this threat, as spyware differ from regular viruses, e.g., in that it uses a different infection technique [10]. Viruses normally replicate themselves by injecting their code into executable files and spread in this way, which is not the case for most spyware.

Specific anti-spyware tools have been developed as countermeasures but there seem to be no single anti-spyware tool that can prevent all existing spyware because, without vigilant examination of a software package, the process of spyware detection has become almost impossible [11]. Current anti-spyware tools make use of signature-based methods by using specific features or unique strings extracted from binaries or heuristic-based methods by using on the basis of rules written by experts who define behavioral patterns as approaches against spyware. These approaches are often considered ineffective against new malicious code [10,12]. Moreover, since most heuristic approaches have been developed in order to detect viruses, it is not certain whether they would be capable of detecting new types of spyware because spyware use stealth systems and they do not employ any specific routines like viruses which may be associated explicitly with spyware [10].

This paper presents a spyware detection method inspired by data mining-based malicious code detection. In this method, binary features are extracted from executable files. A feature reduction method is then used to obtain a subset of data which is further used as a training set for automatically generating classifiers. This method is different from signature-based or heuristic-based methods since no specific matching is performed. In this method, the generated classifiers are used to classify new, previously unseen binaries as either legitimate software or spyware. In our experiments, we employ 10-fold cross validation in order to evaluate classifiers on unseen binaries. We use accuracy and the Area under Receiver Operating Characteristic (ROC) curve as metrics for the evaluation of classifier performance.

II. BACKGROUND

The term spyware first appeared in a Usenet post on October 16, 1995 about a piece of hardware that could be used for espionage. In 2000, the founder of Zone Labs, Gregor Freund, used the term in a press release for Zone Labs' firewall product [13]. Since then, spyware has spread rapidly and several attempts to prevent this spread have been made. In 2001, the use of data mining was investigated as an approach for detecting malware [12] and this attempt attracted the attention of many researchers. Since then, several experiments have been performed to investigate the detection of traditional malicious software such as viruses, worms, and so forth, by using data mining technologies.

The objective of the aforementioned data mining experiment [12] was to detect new and unseen malicious code from available patterns. Data mining is the process of analyzing electronically stored data by automatically searching for patterns [14]. Machine Learning algorithms are commonly used to detect new patterns or relations in data, which are further used to develop a model, i.e., a classifier or

a regression function. Learning algorithms have been used widely for different data mining problems to detect patterns and to find correlations between data instances and attributes. In order to represent malware instances in a suitable format for data mining purposes, many researchers have used n -grams or API calls as their primary type of feature. An n -gram is a sequence of n elements from a population. It can, e.g., represent a character or a word. The length of an n -gram can be either fixed (e.g.: unigrams, bigrams, and trigrams) or variable. In experiments for the detection of malware, sequences of bytes extracted from the hexadecimal dump of the binary files have been represented by n -grams. In addition to the use of such byte sequences, some experimental studies have been conducted using data from End User License Agreements (EULAs), network traffic, and honeypots.

The 2001 data mining study of malicious code [12] used three types of features, i.e., Dynamic-link Library resource information, consecutive printable characters (strings) and byte sequences. The data set consisted of 4,266 files out of which 3,265 were malicious and 1,001 were legitimate or benign programs. A rule induction algorithm called Ripper [15] was applied to find patterns in the DLL data. Naive Bayes (NB), a learning algorithm based on Bayesian statistics [14], was used to find patterns in the string data and n -grams of byte sequences were used as input data for the Multinomial Naive Bayes algorithm [14]. A data set partitioning was performed in which two data sets were prepared, i.e., a test data set and a training data set. This is to allow for performance testing on data that are independent from the data used to generate the classifiers. The Naive Bayes algorithm, using strings as input data, yielded the highest classification performance with an accuracy of 97.11%. The study also implemented a signature-based algorithm and compared its results to those of the data mining algorithms. The data mining-based detection rate of new malware was twice as high in comparison to the signature-based algorithm.

Following this study, a large number of researchers [16,17,18,19,20,21,22] have devoted their efforts for encountering malicious code, which is in most of the cases either viruses or worms, by using data mining. Only two studies [10,23] focused specifically on spyware. References [19, 24, 25] used n -grams of byte code as features while others [17, 20] used opcodes. They all were successful in having more than 97% of accuracy.

In a different study [18], an experiment was performed on network traffic filtered by network scanner but still having suspicious malicious code. Two different types of features were used: n -grams of size 5 and Windows Portable Executable header data. This study was successful in achieving an Area under the ROC curve score of 0.983..

Reference [16] performed an experiment for detection of viruses on a data set of 3,000 files. The study performed experiments on sequence lengths ranging from 3 to 8. The best result was obtained using a sequence length of 5. The results indicated that classifier performance could be increased by using shorter sequences.

Reference [20] performed an experiment for the detection of Trojans. In this study, instruction sequences were used as features. The primary data set contained 4,722 files. Out of these, 3,000 files were Trojans and the rest were benign programs. A detection of compilers, common packer was also performed on data set and the feature set was also systematically reduced. Three types of algorithms were analyzed; Random Forest (RF), Bagging and Decision Trees (DT). The study used ROC as analysis for measuring performance and the best results for false positive rate, overall accuracy and area under the ROC curve were achieved with the Random Forest classifier.

Reference [10] has replicated the work of [12] but with a focus on Spyware collected in 2005. The purpose was to specifically evaluate the suitability of the technique for spyware detection. The data set consisted of 312 benign executables and 614 spyware executables. These spyware applications were not embedded (bundled) with any other executables. The Naive Bayes algorithm was evaluated, using a window size of 2 and 4, with 5-fold cross validation. Cross-validation is a statistical method that is used to systematically divide the available data into a predetermined number of folds, or partitions [14]. Prediction models, or classifiers, are generated by applying a learning algorithm to $n-1$ folds and then evaluated on the n -th fold. The process is repeated until all folds have been used for evaluation once. Even though criticism has been directed towards (over) belief in the cross-validation performance estimates [26], the method is still widely regarded as a reasonable and robust performance estimation method, especially when the data is scarce. The experiment showed that the overall accuracy was higher when using a window size of 4.

The spyware problem is also different from that of detecting viruses or worms as vendors of spyware-hosting applications usually include them in bundles with popular free software. The End User License Agreement (EULA) may very well mention the spyware, in order for the spyware vendors to avoid legal consequences, but this information is given in a way that makes it difficult for the average user to make an informed consent. In addition, the EULAs from both legitimate software vendors and spyware vendors normally contain thousands of words and this makes it hard for users to interpret the information. Reference [23] therefore investigated the possibility to automatically detect spyware by mining the EULA. This study is similar to the studies carried out on spam detection by using data mining. The studied data set contained 996 EULAs out of which 9.6 % were associated with spyware. The study applied 17 learning algorithms on two data sets, represented by a bag-of-words and meta EULA model, respectively. The performances of the 17 classifiers were compared with a baseline classifier, ZeroR, which predicts the class of an instance by always assigning the majority class, e.g., the class that the majority of the instances in the training data set belong to. ZeroR is commonly used as a baseline when evaluating other learning algorithms. The results indicated that the bag-of-words model is better than the meta EULA model. Results also indicated that it is indeed

possible to distinguish between legitimate software and spyware by automatically analyzing the corresponding EULAs.

A majority of the reviewed studies use n -grams to represent byte sequences. Except for Ref. [10] and Ref. [23], all the studies were performed on malware or viruses. Moreover some studies [10, 12, 20] featured data sets with almost a double number or an equal number of malicious files than benign files. Other studies [18, 22] use a population in which a third consists of malicious files. This situation is arguably unrealistic, since in real life, the number of malicious files compared to benign files is much lower. Most of the studies used standard data sets available for malware or virus research. These data sets contain individual malicious executables. Thus, the executables are not embedded or bundled with other executables, which is the common situation for spyware. We have only been able to find one malicious code detection study that focuses on spyware [10]. This study performed experiments using n -grams for data representation, in particular with $n = 2$ and $n = 4$. The latter configuration yielded the best results. However, other experiments on malicious code [18] have showed better results for $n = 5$. We therefore argue that a larger set of n -values need to be evaluated for the spyware domain.

III. PROPOSED METHOD

The focus of our analysis is executable files for the Windows platform. We use the Waikato Environment for Knowledge Analysis (Weka) [14] to perform the experiments. Weka is a suite of machine learning algorithms and analysis tools, which is used in practice for solving data mining problems. First, we extract features from the binary files and we then apply a feature reduction method in order to reduce data set complexity. Finally, we convert the reduced feature set into the Attribute-Relation File Format (ARFF). ARFF files are ASCII text files that include a set of data instances, each described by a set of features [14]. Figure 1 shows the steps involved in our proposed method.

Figure 1: Experimental process

A. Data Collection

Our data set consists of 137 binaries out of which 119 are benign and 18 are spyware binaries. The benign files were collected from Download.com [27], which certifies the files to be free from spyware. The spyware files were downloaded from the links provided by SpywareGuide.com [28], which hosts information about different types of spyware and other types of malicious software. The rather low amount of gathered spyware is attributed to the fact that most of the links

provided by SpywareGuide.com were broken, i.e., these links did not lead to pages where the spyware executables could be downloaded. We have yet to find, or build, a larger spyware data set.

B. Malicious File Percentage

Reference [17] has shown that for their particular study, the MFP needed to be equal to, or lower than, 15% of the total population in order to yield a high prediction performance. Relating to our data set, the MFP is almost 14%. However, it is important to stress that we have yet to uncover evidence to support that the recommended MFP leads to improved results in the general case.

C. Byte Sequence Generation

We have opted to use byte sequences as data set features in our experiment. These byte sequences represent fragments of machine code from an executable file. We use xxd [29], which is a UNIX-based utility for generating hexadecimal dumps of the binary files. From these hexadecimal dumps we may then extract byte sequences, in terms of n -grams of different sizes.

D. n -gram Size

A number of research studies have shown that the best results are gained by using an n -gram size of 5 [16,18]. In the light of previous research, we chose to evaluate three different n -gram sizes (namely: 4, 5, and 6) for the experiments.

E. Parsing

We first extract byte sequences of the desired n -size. Each row contains one n -gram and the length of a single row is thus equal to the size of n .

F. Feature Extraction

The output from the parsing is further subjected to feature extraction. We extract the features by using two different approaches: the Common Feature-based Extraction (CFBE) and the Frequency-based Feature Extraction (FBFE). The purpose of employing two approaches is to evaluate two different techniques that use different types of data representation, i.e., the occurrence of a feature and the frequency of a feature. Both methods are used to obtain Reduced Feature Sets (RFSs) which are then used to generate the ARFF files.

1. Common Feature-based Extraction

In CFBE, the common n -grams (byte sequences) are extracted from the binary files, one class at a time.

2. Frequency-based Feature Extraction

The word frequency can be defined in various ways. In statistics it basically represents the number of occurrences or repetitions of some observation at a specific time or from some specific category. In our study, the word frequency means the number of occurrences of some specific n -gram in a certain class or the number of repetitions of some specific n -

gram in a particular class. In FBFE, all the n -grams were sorted and the frequency of each n -gram in each class is calculated. All n -grams, within a specified frequency range, are extracted and the rest are discarded. In the frequency calculation, we discovered that there were a few uninteresting n -grams, e.g.: 0x0000000000, 0xFFFFFFFFFFFF, and 0x0000000001. Even though these instances were few (less than 10), their frequencies were high (more than 10,000). Thus, the frequency analysis helped us to define three suitable frequency ranges: 1-49, 50-80, and 81-500. The number of n -grams in the 50-80 frequency range tends to be almost equal to the number of n -grams in the 81-500 range.

G. Feature Reduction

Features were reduced in both the CBFE and FBFE method. In CBFE, the common features gained from all files were sorted. Only one representation of each feature was considered in one class. CBFE has produced a better reduced feature set. For example, the reduced feature set for $n = 4$ contains only 536 features compared to 34,832,131 for the complete set.

In FBFE, the frequency of each n -gram is calculated. Reduced features were obtained with three frequency ranges 1-49, 50-80, and 81-500. After analysis of the number of n -grams in each frequency range, it was decided that the 1-49 frequency range will not be included in the experiments, since the number of n -grams even in the reduced feature set was too high. For example, for an n -gram size of 5, the total number of n -grams was 27,865,739 and the number of n -grams included in the reduced feature set for the 1-49 frequency range was 21,987,533, which indicated the presence of a large amount of uninteresting features.

Table 1. shows some statistics regarding the number of features in each class, the total number of features, and the reduced feature sets, based on different frequency ranges for both FBFE and CFBE.

Size / Features	$n = 4$	$n = 5$	$n = 6$
Benign Features	26,474,673	21,179,768	17,649,809
Spyware Features	8,357,458	6,685,971	5,571,645
Total Features	34,832,131	27,865,739	23,221,454
FR = 1 - 49	26,269,292	21,987,533	18,746,618
FR = 50 - 80	5,282	3,226	2,286
FR = 81 - 500	6,018	3,929	2,788
CFBE	536	514	322

FR= Frequency Range

H. ARFF Generation

Two ARFF databases based on frequency and common features were generated. All input attributes in the data set are represented by Booleans, i.e., either a certain n -gram or the n -grams within a certain frequency range are represented by either 1 or 0 (present or absent).

I. Classifiers

Previous studies are not conclusive about which learning algorithm generates the best classifiers for problems similar to

Table 2: Accuracy results of experiment

n	Method	ZeroR	NaiveBayes	SMO	J48	RandomForest	JRip
4	CBFE	86.92 (2.72)	88.20(6.17)	86.56(8.13)	89.89(5.10)	89.48(5.52)	89.45(4.94)
	FR 50_80	86.92 (2.72)	83.70(8.20)	88.52(7.53)	88.52(5.89)	87.07(6.47)	88.07(6.28)
	FR 81_500	86.92 (2.72)	79.37(12.28)	89.65(5.93)	88.41(6.21)	88.82(5.66)	85.84(7.31)
5	CBFE	86.92(2.72)	89.36(5.43)	83.41(8.59)	89.88(4.78)	88.61(6.19)	89.15(4.36)
	FR 50_80	86.92(2.72)	87.25(6.19)	86.39(6.65)	87.15(7.72)	88.45(5.93)	88.21(5.85)
	FR 81_500	86.92(2.72)	87.39(5.69)	88.42(5.96)	84.68(6.97)	87.94(5.57)	85.95(7.17)
6	CBFE	86.92 (2.72)	89.80(4.89)	87.37(6.98)	90.54(4.33)	88.74(5.75)	88.77(5.40)
	FR 50_80	86.92 (2.72)	87.41(6.26)	88.02(6.10)	88.30(5.51)	89.30(5.45)	88.15(4.21)
	FR 81_500	86.92 (2.72)	88.04(6.13)	89.29(7.26)	86.96(7.50)	88.25(6.01)	86.98(7.37)

FR= Frequency Range

the studied problem. However, the results have provided us with a basis for choosing ZeroR, Naive Bayes, Support Vector Machines (SMO), C4.5 Decision Tree (J48), Random Forest and JRip as candidates for our study. ZeroR is used only as a baseline for comparison. For our purpose, it can be viewed as a random guesser, modeling a user that makes an uninformed decision about a piece of software. A Naive Bayes classifier is a probabilistic classifier based on Bayes theorem with independence assumptions, i.e., the different features in the data set are assumed not to be dependent of each other. This of course, is seldom true for real-life applications. Nevertheless, the algorithm has shown good performance for a wide variety of complex problems. SVMs, which are used for classification and regression, work by finding the optimal hyperplane, which maximizes the distance / margin between two classes. J48 is a decision tree-based learning algorithm. During classification, it adopts a top-down approach and traverses a tree for classification of any instance. Moreover, Random Forest is an ensemble learner. In this ensemble, a collection of decision trees are generated to obtain a model that may give better predictions than a single decision tree. Meanwhile, JRip is a rule-based learning algorithm.

J. Performance Evaluation Criteria

We evaluate each learning algorithm by performing cross-validation tests to ensure that the generated classifiers are not tested on the training data. From the response of the classifiers the relevant confusion matrices were created. Four metrics define the elements of the matrix: True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). TP represents the correctly identified benign programs while FP represents the incorrectly classified spyware programs. Correspondingly, TN represents the correctly identified spyware programs and FN represents the wrongly identified benign programs.

The performance of each classifier was evaluated using the AUC metric and the (overall) Accuracy (ACC) metric. The later is defined in Equation 3. AUC is essentially a single-point value derived from a ROC curve, which is commonly used when the performance of a classifier needs to be evaluated for the selection of a high proportion of positive instances in the data set [14]. Therefore it plots the True Positive Rate (TPR, see Equation 1) on the x -axis in function of the False Positive Rate (FPR, see Equation 2) on the y -axis at different points. TPR is the ratio of correctly identified benign programs while FPR is the ratio of wrongly identified Spyware programs. ACC is the percentage of correctly

identified programs. In many situations, ACC can be a reasonable estimator of performance (the performance on yet unseen data). However, AUC has the benefits of being independent of class distribution and cost [30]. In many real-world problems, the classes are not equally distributed and the cost of misclassifying one class may be different to that of misclassifying another. For such problems, the ACC metric is not a good measure of performance but it may be used as a complementary metric.

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

$$FPR = \frac{FP}{TN + FP} \quad (2)$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

IV. RESULTS

Table 2 and Table 3 show the results for each n -gram size for both the CFBE and the FBFE method. Two feature sets were produced as a result of the FBFE approach. The first feature set includes instances from the frequency range of 50-80 and the second set includes instances from the 81-500 frequency range. Each table shows the results of the baseline classifier and five different learning algorithms. As stated earlier, we represent classification performance using two metrics: AUC and ACC. Algorithms that are significantly better or worse than the baseline in terms of AUC according to the corrected paired t-test (confidence 0.05, two-tailed) are also indicated. It is not the main objective of this study to determine the optimal algorithm or the highest possible performance. Hence, we did not tune the parameters of the learning algorithms, i.e., all algorithms are default configured.

A. Results for $n = 4$

Using the feature set produced by the CFBE feature selection method for $n = 4$, the J48 decision tree classifier achieves the highest accuracy results. However, it only performs slightly better than the Random Forest model and the JRip classifier. In comparison, the accuracy of Naive Bayes is mediocre while the support vector machines classifier (SMO) achieved the lowest accuracy. In summary, all included algorithms have performed better than the baseline. When comparing the AUC-based performance results, the Random Forests model achieved the highest score while J48 performed at a mediocre level.

Table 3: Area Under ROC Curve results of experiment

n	Method	ZeroR	NaiveBayes	SMO	J48	RandomForest	JRip
4	CBFE	0.50(0.00)	0.61(0.17)*	0.63(0.18)*	0.63(0.16)*	0.73(0.23)	0.61(0.15)
	FR 50_80	0.50(0.00)	0.62(0.20)	0.71(0.18)	0.62(0.16)	0.78(0.20)	0.60(0.16)
	FR 81_500	0.50(0.00)	0.62(0.23)	0.67(0.19)*	0.61(0.17)	0.78(0.20)*	0.56(0.14)
5	CBFE	0.50(0.00)	0.62(0.17)*	0.58(0.16)	0.63(0.15)*	0.68(0.22)*	0.59(0.13)*
	FR 50_80	0.50(0.00)	0.61(0.17)	0.66(0.17)*	0.64(0.20)*	0.76(0.23)*	0.61(0.16)*
	FR 81_500	0.50(0.00)	0.61(0.16)*	0.67(0.17)*	0.62(0.24)	0.78(0.21)*	0.66(0.19)*
6	CBFE	0.50(0.00)	0.61(0.19)	0.62(0.17)	0.63(0.16)	0.75(0.22)	0.61(0.16)
	FR 50_80	0.50(0.00)	0.60(0.16)	0.66(0.17)*	0.59(0.15)	0.82(0.17)*	0.57(0.12)
	FR 81_500	0.50(0.00)	0.62(0.18)*	0.71(0.20)*	0.65(0.18)*	0.83(0.17)*	0.66(0.18)*

*Significantly better at confidence 0.05 two tailed
FR= Frequency Range

Figure 2: Comparison of Accuracy with $n = 6$ Figure 3: Comparison of Area under ROC Curve with $n = 6$

In the 50-80 frequency range, SMO and J48 produced the highest ACC results. Moreover, in the higher range (81-500), SMO yields the highest ACC-based performance. In the frequency range of 50-80, SMO and J48 are slightly better than JRip and in the frequency range of 81-500; SMO is performing slightly better than J48. When comparing results across these two frequency ranges, it is obvious that the difference in accuracy is negligible.

B. Results for $n = 5$

Similarly to the results for $n = 4$, the feature set produced by the CFBE feature selection method was suitable as a data representation for the J48 algorithm, which again yielded the best ACC results. In comparison, the SMO produced the worst ACC result of all included algorithms for this particular data set. In fact, the results of the SMO are even lower than the baseline. In contrast, the difference between the results of J48, Naive Bayes and JRip is small and all algorithms performed better than the baseline. In terms of the AUC, the Random Forest algorithm was the best performer, followed by the J48.

For the 50-80 frequency range, the Random Forest algorithm yielded the highest ACC results. However, it only slightly outperformed the JRip rule inducer. J48 performed mediocre at this range. Moreover, in the 81-500 range, the SMO yielded the highest ACC results while NB and RF were mediocre. In terms of the area under the ROC curve, the Random Forest algorithm outperformed the other algorithms for both the 50-80 and the 81-500 range. When comparing results obtained for both of these frequency results, the 50-80 frequency range seems to be more suitable on average than the 81-500 frequency range.

C. Results for $n = 6$

The data sets generated for $n = 6$ proved to be most successful in terms of both accuracy and the area under the ROC curve. The feature set produced by the CFBE feature selection algorithm was used in conjunction with the J48 decision tree algorithm to yield a top accuracy score of 90.5%. It slightly outperformed the NB algorithm, which was followed by RF and JRip. The support vector machines algorithm yielded the lowest ACC score. All algorithms performed better than the baseline. For the AUC Random Forest was the best performer, yielding a top AUC score of 0.83.

V. DISCUSSION

The feature sets generated by the CFBE feature selection method generally produced better results with regard to accuracy than the feature sets generated by the FBFE feature selection method. However, the reverse situation seems to be true when AUC is used as an evaluation metric. Overall, the results suggest that the two higher frequency ranges are more suitable than the lowest. The best AUC result was obtained by using the Random Forest algorithm, an n -gram size of 6, and the highest frequency range. However, the 50-80 frequency range yielded better AUC results than the other frequency ranges when the size of n -grams was 4. When comparing the performance on the data sets generated by the FBFE method, it is clear that the ACC results for the 50-80 frequency range are better than those for the 81-500 frequency range. This can easily be viewed in Figure 2. Meanwhile, the AUC results are very close to each other for both of these ranges as shown in Figure 3. Consequently, more experiments, e.g., with larger amounts of data and a wider variety of learning algorithms are

needed in order to fully understand which data representation and feature selection method is optimal for our purpose.

VI. CONCLUSIONS AND FUTURE WORK

Data mining-based malicious code detectors have been proven to be successful in detecting clearly malicious code, e.g., like viruses and worms. Results from different studies have indicated that data mining techniques perform better than traditional techniques against malicious code. However, spyware has not received the same attention from researchers but it is spreading rapidly on both home and business computers. The main objective of this study was therefore to determine whether spyware could be successfully detected by classifiers generated from n -gram based data sets, which is a common data mining-based detection method for viruses and other malicious code.

In order to find a suitable data representation, we analyzed n -gram-based byte sequences of different sizes from a range centered on $n = 5$, which has proven to be an appropriate value that yields high performance for similar experiments. We then evaluated five common learning algorithms by generating classifiers and using 10-fold cross validation and the corrected paired t-test. Moreover, two different feature selection methods were compared for all algorithms and n -gram sizes. Since no suitable spyware data set was available, we collected spyware and legitimate binaries and generated a small data set for the purpose of validating our approach. The experiments indicate that the approach is successful, achieving a 90.5 % overall accuracy with the J48 decision tree algorithm when using $n = 6$ and the common n -gram feature selection method. The success of the approach is also indicated by an AUC score of 0.83 with the Random Forest algorithm when using $n = 6$ and the frequency-based feature selection method. Currently, the false positive rate is quite high for most combinations of algorithms and data sets. However, we believe that one of the primary reasons for this is that the data set is small. In particular the number of spyware files is too low. In data mining, it is believed that larger set of data can produce better results [14]. So a larger set of data can be tested to have better classification with higher ACC and lower false positive rate. With regard to AUC, which is our primary evaluation metric, all algorithms were statistically significantly better than the baseline but with different combinations of n -gram size and feature selection method. Thus, from our experiments, we can conclude that it is possible to detect spyware by using automatically generated classifiers to identify patterns in executable files.

We hope that data mining techniques can help the researcher community and security experts to label the software or the home user to have an informed decision before installation of any software. For future work, we plan to gather a larger collection of binary files, especially spyware binaries as no standard data set of spyware is currently available, and to evaluate our approach when the data set features represent opcodes instead of arbitrary bytes. Additionally, we aim to develop a hybrid spyware

identification method that is based on a combination of EULA-based and executable-based detection techniques.

ACKNOWLEDGMENT

The authors first and foremost thank Dr. Bengt Carlsson of Blekinge Institute of Technology, Sweden, for providing his expertise and advice. The authors also thank the anonymous reviewers for their time and helpful comments.

REFERENCES

- [1] M. Boldt and B. Carlsson, "Privacy-invasive software and preventive mechanisms," *2nd International Conference on Systems and Networks Communications, (ICSNC 2006)*, Oct. 28- Nov.2, IEEE Computer Society.
- [2] M. Wu, Y. Huang, S. Kuo, "Examining Web-based spyware invasion with stateful behavior monitoring," *13th Pacific Rim International Symposium on Dependable Computing (PRDC '07)*, 17-19 Dec. Piscataway, NJ, USA: IEEE, pp. 275-81.
- [3] R. Sandhu, "Lattice-based access control models," *Computer*, vol. 26, Nov. 1993, pp. 9-19.
- [4] R. Stern, "FTC cracks down on spyware and PC hijacking, but not true lies," *Micro*, vol. 25, 2005, pp. 100-101.
- [5] N. Arastouie and M. Razzazi, "Hunter: an anti spyware for windows operating system," *3rd International Conference on Information and Communication Technologies: from Theory to Applications, (ICTTA)*, , 2008, pp. 1-5.
- [6] Spyware, us-cert.gov/reading_room/Spywarehome_0905.pdf [accessed 2009-05-18].
- [7] T. Bollinger, "Software in the year 2010," *IT Professional*, vol. 6, 2004, pp. 11-15.
- [8] H. Qing and T. Dinev, "Is spyware an internet nuisance or public menace?" *Communications of the ACM*, vol. 48, 2005, pp. 61-66.
- [9] W. Ames, "Understanding spyware: Risk and response," *IT Professional*, vol. 6, 2004, pp. 25-29.
- [10] C. D. Bozagac, "Application of Data Mining based Malicious Code Detection Techniques for Detecting new Spyware", White paper, Bilkent University, 2005.
- [11] M. Wu, Y. Huang, Y. Wang, and S. Kuo, "A stateful approach to spyware detection and removal," *12th Pacific Rim International Symposium on Dependable Computing, (PRDC 2006)*, 18-20 Dec. Los Alamitos, CA, USA: IEEE Computer Society, pp.173-182.
- [12] M. Schultz, E. Eskin, F. Zadok, and S. Stolfo, "Data mining methods for detection of new malicious executables," *Proceedings of IEEE Symposium on Security and Privacy, 14-16 May 2001*, Los Alamitos, CA, USA: IEEE Computer Society, pp. 38-49.
- [13] Zone Alarm, Press Release 2000, <http://www.zonealarm.com> [accessed 2009-05-14].
- [14] I.H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan

- Kaufmann, 2005.
- [15] W. Cohen, "Fast effective rule induction," *Proc. 12th International Conference on Machine Learning*, pp. 115-23, San Francisco, CA: Morgan Kaufmann Publishers, 1995.
 - [16] O. Henchiri and N. Japkowicz, "A feature selection and evaluation scheme for computer virus detection," *6th International Conference on Data Mining (ICDM'06), 18-22 Dec.*, Piscataway, NJ, USA: IEEE, pp. 918-922.
 - [17] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, S. Dolev, and Y. Elovici, "Unknown malware detection using OPCODE representation," *1st European Conference on Intelligence and Security Informatics, (EuroISI 2008), 3-5 Dec.*, Berlin, Germany: Springer-Verlag, pp. 204-215.
 - [18] Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer, "Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic", *Proceedings of the 30th annual German conference on Advances in Artificial Intelligence, KI 2007, 10-13 Sept.*, Berlin, Germany: Springer-Verlag, pp. 44-50.
 - [19] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," *Proceedings of the 28th Annual International Computer Software and Applications Conference. COMPSAC 2004, 28-30 Sept.*, Los Alamitos, CA, USA: IEEE Computer Society, pp. 41-42.
 - [20] M. Siddiqui, M. Wang, and J. Lee, "Detecting Trojans Using Data Mining Techniques," *Wireless Networks, Information Processing and Systems*, 2009, pp. 400-411.
 - [21] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu, "Virus detection using data mining techniques," *Proceedings of International Carnahan Conference on Security Technology, 14-16 Oct. 2003*, Piscataway, NJ, USA: IEEE, pp. 71-76.
 - [22] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, and Y. Elovici, "Unknown malware detection via text categorization and the imbalance problem," *International Conference on Intelligence and Security Informatics, (ISI 2008), 17-20 June*, Piscataway, NJ, USA: IEEE, pp. 156-161.
 - [23] N. Lavesson, M. Boldt, P. Davidsson and A. Jacobsson, "Learning to Detect Spware using End User License Agreements," *Knowledge and Information Systems*, in press.
 - [24] J.Z. Kolter and M.A. Maloof, "Learning to detect malicious executables in the wild", *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining, KDD 2004, 22- 25 Aug.*, Seattle, WA, United states: ACM, pp. 470-478.
 - [25] D. Reddy, S. Dash, and A. Pujari, "New Malicious Code Detection Using Variable Length n-grams," *Information Systems Security*, 2006, pp. 276-288.
 - [26] Isaksson, M. Wallman, H. Göransson, and M. Gustafsson, "Cross-validation and bootstrapping are unreliable in small sample classification," *Pattern Recognition Letters*, vol. 29, Oct. 2008, pp. 1960-1965.
 - [27] Download, <http://download.com> [accessed 2009-05-17].
 - [28] Spyware Guide, <http://Spywareguide.com> [accessed 2009-04-23].
 - [29] Linux / Unix Command: xxd, http://linux.about.com/library/cmd/blcmdl1_xxd.htm [accessed 2009-04-26].
 - [30] F. Provost, T. Fawcett and R. Kohavi, "The Case against Accuracy Estimation for Comparing Induction Algorithms". *Proceedings of the Fifteenth international Conference on Machine Learning, ICML 1998, 24-27 July*, San Francisco, CA, USA: Morgan Kaufmann Publishers, pp. 445-453.