Feature Based Techniques for Auto-Detection of Novel Email Worms

Mohammad M Masud, Latifur Khan, and Bhavani Thuraisingham

Department of Computer Science
The University of Texas at Dallas
Richardson, Texas-75083
{mehedy, lkhan, bhavani.thuraisingham}@utdallas.edu

Abstract. This work focuses on applying data mining techniques to detect email worms. We apply a feature-based detection technique. These features are extracted using different statistical and behavioral analysis of emails sent over a certain period of time. The number of features thus extracted is too large. So, our goal is to select the best set of features that can efficiently distinguish between normal and viral emails using classification techniques. First, we apply Principal Component Analysis (PCA) to reduce the high dimensionality of data and to find a projected, optimal set of attributes. We observe that the application of PCA on a benchmark dataset improves the accuracy of detecting novel worms. Second, we apply J48 decision tree algorithm to determine the relative importance of features based on information gain. We are able to identify a subset of features, along with a set of classification rules that have a better performance in detecting novel worms than the original set of features or PCA-reduced features. Finally, we compare our results with published results and discuss our future plans to extend this work.

Keywords: Email worm, data mining, feature selection, Principal Component Analysis, classification technique.

1 Introduction

Worms are malicious code that infect a host machine and spread copies of itself through the network to infect other hosts. There are different kinds of worms such as: Email worms, Instant Messaging worms, Internet worms, IRC worms and File-sharing Networks worms. Email worm, as the name implies, spreads through infected email messages. The worm may be carried by attachment, or the email may contain links to an infected website. When the user opens the attachment, or clicks the link, the host is immediately infected. Email worms use the vulnerability of the email software at the host machine and sends infected emails to the addresses stored in the address book. In this way, new machines get infected. Examples of email worms are "W32.mydoom.M@mm", "W32.Zafi.d", "W32.LoveGate.w", "W32.Mytob.c", and so on. Worms do a lot of harm to computers and people. They can clog the network traffic, cause damage to the system and make the system unstable or even unusable.

Z.-H. Zhou, H. Li, and Q. Yang (Eds.): PAKDD 2007, LNAI 4426, pp. 205–216, 2007.

There has been a significant amount of research going on to combat worms. The traditional way of dealing with a known worm is to apply signature based detection. Once a new worm appears, researchers work hard to find a unique pattern in the code that can identify it as a particular type of worm. This unique pattern is called the *signature* of the worm. Thus, a worm can be detected from its signature. But the problem with this approach is that it involves significant amount of human intervention and it may take long time (from days to weeks) to discover the signature. Since worms can propagate very fast, there should be a much faster way to detect them before any damage is done.

We are concerned with the problem of detecting new email worms without knowing their signatures. Thus, our work is directed towards automatic (i.e., without any human intervention) and efficient detection of novel worms. Our work is inspired by [1], which does not require signature detection; rather, it extracts different features from the email (to be explained shortly) and tries to classify the email as clean or infected. They employ two classifiers in series, the first one is Support Vector Machine (SVM) and the next one is Naïve Bayes (NB). We will refer to this two-layer approach as 'SVM + NB series' approach. They report a very high accuracy, as well as very low false positive and false negative rate. But we address several issues with their approach. First, they do not apply a balanced dataset to test classification accuracy. Second, they do not apply cross validation on the dataset. Third, our experimental results indicate that a two-layer approach is less efficient than a single layer approach in terms of cross validation accuracy on a balanced dataset. Fourth, they apply too many features, most of which is found to be redundant by our study. We deal with all these problems and provide efficient solutions.

Our contributions to this research work are as follows: First, we compare individual performances of NB and SVM with their SVM + NB series counterpart. We show that the series approach is less effective than either NB or SVM alone. So, we claim that one of the layers of the two-layer approach is redundant. Second, we rearrange the dataset so that it becomes more balanced. We divide the dataset into two portions: one containing only known worms (and some clean emails), the other containing only a novel worm. Then we apply a three-fold cross validation on the 'known worm' dataset, and we test the accuracy of each of the learned classifiers on the 'novel worm' dataset. We report the cross validation accuracy, and novel detection accuracy for each of the datasets. Third, we apply PCA on the dataset to improve the efficiency of classification task. PCA is commonly used to extract patterns from high dimensional data, especially when the data is noisy. Besides, it is a simple and nonparametric method. Since the original dataset contains a total of 94 attributes, it is very likely that some of these attributes are redundant, while some others add noise into the data, and PCA could be effectively applied to reduce this data to a lower dimension; revealing the underlying simple pattern hidden in the data. Fourth, we build decision tree, using the WEKA [2] implementation of C4.5 [3] called the J48, from the dataset to identify the most important features according to information gain. We find that only a few features are sufficient to obtain similar or better classification accuracy. We report the features as well as the classification rules obtained from the decision tree.

The rest of this paper is organized as follows: section 2 describes related work in automatic email worm detection, section 3 describes the feature reduction and

selection techniques, section 4 describes the dataset, section 5 describes the experiments, section 6 discusses the results, and section 7 concludes with future guidelines for research.

2 Related Work

There are different approaches to automate the detection of worms. These approaches are mainly of two types: behavioral and content-based. Behavioral approaches analyze the behavior of messages like source-destination addresses, attachment types, message frequency etc. Content-based approaches look into the content of the message, and try to detect signature automatically. There are also combined methods that take advantage of both techniques.

An example of behavioral detection is social network analysis [4, 5]. It detects worm infected emails by creating graphs of network, where users are represented as nodes, and communications between users are represented as edges. A social network is a group of nodes among which there exists edges. Emails that propagate beyond the group boundary are considered to be infected. But the drawback of this system is that worms can easily bypass social networks by intelligently choosing the recipient lists, by looking at recent emails in the user's outbox.

Statistical analysis of outgoing emails is another behavioral approach [6, 7]. Statistics collected from frequency of communication between clients and their mail server, byte sequences in the attachment etc. are used to predict anomalies in emails and thus worms are detected.

Example of content based approach is the EarlyBird System [8]. In this system, statistics on highly repetitive packet contents are gathered. These statistics are analyzed to detect possible infection of host or server machines. This method generates content signature of worm without any human intervention. Results reported by this system indicated very low false positive rate of detection. Other examples are the Autograph [9], and the Polygraph [10], developed at Carnegie Mellon University.

There are other approaches to detect early spreading of worms, such as employing honeypot" [11]. A honeypot is a closely monitored decoy computer that attracts attacks for early detection and in-depth adversary analysis. The honeypots are designed to not send out email in normal situations. If a honeypot begins to send out emails after running the attachment of an email, it is determined that this email is an email worm.

Martin et al. [12] also report an experiment with email data, where they apply a statistical approach to find an optimum subset of a large set of features to facilitate the classification of outgoing emails, and eventually, detect novel email worms.

Another approach by Sidiroglou et al. [13] employs behavior-based anomaly detection, which is different from the signature based or statistical approaches. Their approach is to open all suspicious attachments inside an instrumented virtual machine looking for dangerous actions, such as writing to the Windows registry, and flag suspicious messages.

Although our approach is feature-based, it is different from the above feature-based detection approaches in that we apply PCA, and decision tree to reduce the dimension

of data. Rather than choosing a subset of features, PCA finds a linear combination of the features and projects them to a lower dimension, reducing noise in data. On the other hand, we apply decision tree to identify the most important features, thereby removing redundant or noisy features. Both these approaches achieve higher accuracy.

3 Feature Reduction and Classification Techniques

Firstly, we briefly describe the features that are used in email worm detection. These features are extracted from a repository of outgoing emails collected over a period of two years [1]. These features are categorized into two different groups: i) per-email feature and ii) per-window feature. Per-email features are features of a single email, while per-window features are features of a collection of emails sent within a window of time. Secondly, we describe our feature reduction techniques, namely, PCA and J48. Finally, we briefly describe the two-layer approach and its limitations.

3.1 Feature Description

For a detailed description of the features please refer to [12]. Each of these features are either continuous valued or binary. Value of a binary feature is either 0 or 1, depending on the presence or absence of this feature in a data point. There are a total of 94 features. Here we describe some of them.

3.1.1 Per Email Features

i. *HTML in body*: Whether there is HTML in the email body. This feature is used because a bug in the HTML parser of the email client is a vulnerability that may be exploited by worm writers. It is a binary feature.
ii. *Embedded image*: Whether there is any embedded image. This is used because a buggy image processor of the email client is also vulnerable to attacks. It is a binary feature.
iii. *Hyperlinks*: Whether there are hyperlinks in the email body. Clicking an infected link causes the host to be infected. It is also a binary feature.
iv. *Binary Attachment*: Whether there are any binary attachments. Worms are mainly propagated by binary attachments. This is also a binary feature.
v. *Multipurpose Internet Mail Extensions* (*MIME*) *type of attachment*: There are different MIME types, for example: "application/msword", "application/pdf", "image/gif", "text/plain" etc. Each of these types is used as a binary feature (total 27).
vi. *UNIX "magic number" of file attachments*: Sometimes a different MIME type is assigned by the worm writers to evade detection. Magic numbers can accurately detect the MIME type. Each of these types is used as a binary feature (total 43).
vii. *Number of attachments*: It is a continuous feature.
viii. *Number of words/characters in subject/body*: These features are continuous. Most worms choose random text, whereas a user may have certain writing characteristics. Thus, these features are sometimes useful to detect infected emails.

3.1.2 Per Window Features

i. *Number of emails sent in window*: An infected host is supposed to send emails at a faster rate. This is a continuous feature.
ii. *Number of unique email recipients*, *senders*: These are also important criteria to distinguish between normal and infected host. This is a continuous feature too.
iii. *Average number of words/characters per subject, body; average word length*: These features are also useful in distinguishing between normal and viral activity.
iv. *Variance in number of words/characters per subject, body; variance in word length:* These are also useful properties of email worms.
v. *Ratio of emails with attachments:* usually, normal emails do not contain attachments, whereas most infected emails do contain them.

3.2 Feature Reduction and Selection

The high dimensionality of data always appears to be a major problem for classification tasks because i) it increases the running time of the classification algorithms, ii) it increases chance of overfitting, and iii) large number of instances are required for learning tasks. We apply PCA to obtain a reduced dimensional data and apply decision tree to select a subset of features, in order to eliminate these problems.

3.2.1 Principal Component Analysis: Reducing Data Dimension

PCA finds a reduced set of attributes by projecting the original attributes into a lower dimension. We observe that for some optimal dimension of projection, the reduced dimensional data observes a better accuracy in detecting novel worms. PCA not only reduces the dimension of data to eliminate all these problems, but also discovers hidden patterns in data, thereby increasing classification accuracy of the learned classifiers. As high dimensional data contains redundancies and noises, it is much harder for the learning algorithms to find a hypothesis consistent with the training instances. The learned hypothesis is likely to be too complex and susceptible to overfitting. PCA reduces the dimension, without losing much information, and thus allows the learning algorithms to find a simpler hypothesis that is consistent with the training examples, and thereby reduces the chance of overfitting. But it should be noted that PCA projects data into a lower dimension in the direction of maximum dispersion. Maximum dispersion of data does not necessarily imply maximum separation of between – class data and/or maximum concentration of within – class data. If this is the case, then PCA reduction may result in poor performance. That is why we apply PCA to reduce the 94-dimensional data into different lower dimensions, ranging from 5 to 90, and select the optimal dimension that achieves the highest classification accuracy.

3.2.2 Decision Tree: Feature Selection Using Information Gain

Feature selection is different from dimension reduction because it selects a subset of the feature set, rather than projecting combination of features onto lower dimension. There are different feature selection techniques available, such as greedy selection, which selects the features one after another until the classification accuracy deteriorates. But the problem with this selection approach is that it takes a lot of time and depending on the order of selection, results vary significantly. We apply the

decision tree approach for feature selection because of three reasons. First, it is fast, second, it applies information gain to select best features, and finally, we can extract a set of rules from the decision tree that reveals the true nature of the 'positive' or the 'negative' class. Thus, we are not only aware of the essential attributes but also get an overall idea about the infected emails. It may also be possible to generalize two or more rules to obtain a generalized characteristic of different types of worms.

Information gain is a very effective metric in selecting features. Information gain can be defined as a measure of the effectiveness of an attribute (i.e., feature) in classifying the training data [14]. If we split the training data on this attribute values, then information gain gives the measurement of the expected reduction in entropy after the split. The more an attribute can reduce entropy in the training data, the better the attribute in classifying the data. Information Gain of a binary attribute A on a collection of examples S is given by (1):

$$Gain\ (S, A) \equiv Entropy\ (S) - \sum_{V \in Values\ (A)} \frac{|\ S_v\ |}{|\ S\ |} Entropy\ (S_v) \tag{1}$$

Where $Values(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v. In our case, each binary attribute has only two possible values (0, 1). Entropy of subset S is computed using the following equation:

$$Entropy(S) = -\frac{p(s)}{n(s)+p(s)} \log_2\left(\frac{p(s)}{n(s)+p(s)}\right) - \frac{n(s)}{n(s)+p(s)} \log_2\left(\frac{n(s)}{n(s)+p(s)}\right) \tag{2}$$

Where $p(S)$ is the number of positive examples in S and $n(S)$ is the total number of negative examples in S. Computation of information gain of a continuous attribute is a little tricky, because it has infinite number of possible values. One approach followed by Quinlan [3] is to find an optimal threshold, and split the data into two halves. The optimal threshold is found by searching a threshold value with highest information gain within the range of values of this attribute in the dataset.

We use J48 for building decision tree, which is an implementation of C4.5. Decision tree algorithms choose the best attribute based on information gain criterion at each level of recursion. Thus, the final tree actually consists of the most important attributes that can distinguish between the positive and negative instances. The tree is further pruned to reduce chances of overfitting. Thus, we are able to identify the features that are necessary and the features that are redundant, and use only the necessary features. Surprisingly enough, in our experiments we find that only four/five features are necessary among the ninety-four features. The decision trees generated in our experiments have better classification accuracies than both original and PCA reduced data.

3.3 Classification Techniques

We apply the NB [15] and SVM [16] classifiers in our experiments. We also apply the SVM+NB series classifier and J48 Decision Tree classifier. NB, SVM and the series classifiers are applied on the original data and the PCA-reduced data, while the J48 is applied on the original data only, because the classifier itself selects a subset of features, discarding redundant ones. The SVM+NB series is implemented as per [1].

We do not recommend using the series classifier because of the following reasons. First, it is not practical. Because we must come up with a set of parameter values such that the false positive of SVM becomes zero. Given a set of continuous parameters, the problem of finding this optimal point is computationally intractable. Second, the assumption in this approach is wrong. Because, even if we happen to find an optimal point luckily, there is no guarantee that these set of values will work on a test data, since this optimal point is obtained from the training data. Third, if NB performs poorly on a particular test set, the output would also be poor. Because, both NB and SVM must perform well to produce a good series result, if any one fails, the combined approach would also fail. In our experimental results, we have indicated the effect of all these problems.

4 Dataset

We have collected the worm dataset used in the experiment by Martin et al. [1]. They have accumulated several hundreds of clean and worm emails over a period of two years. All these emails are outgoing emails. Several features are extracted from these emails as explained in section 3.1.

There are six types of worms contained in the dataset: VBS.BubbleBoy, W32.Mydoom.M, W32.Sobig.F, W32.Netsky.D, W32.Mydoom.U, and W32.Bagle.F. But the classification task is binary: {clean, infected}. The original dataset contains six training and six test sets. Each training set is made up of 400 clean emails and 1000 worm emails. The worm emails are made up of 200 examples from each of the five different worms. The sixth virus is then included in the test set, which contains 1200 clean emails and 200 infected messages.

5 Experiments

As we have mentioned earlier, the dataset is imbalanced. So we apply both cross validation and novel worm detection in our experiments. In our distribution, each balanced set contains 1600 clean email messages, which are the combination of all the clean messages in the original dataset (400 from training set, 1200 from test set). Also, each balanced set contains 1000 viral messages (from original training set), marked as “known worms” and 200 viral messages (the sixth worm from the original test set), and marked as “novel worm”. The cross validation is done as follows: we randomly divide the set of 2600 (1600 clean + 1000 viral) messages into three equal sized subsets. We take two subsets as training set and the remaining set as the test set. This is done three times by rotating the testing and training sets. We take the average accuracy of these three runs. This accuracy is shown under the column *accuracy* in following tables. Besides testing the accuracy of the test set, we also test the detection accuracy of the learned classifier on the “novel worm” set. This accuracy is also averaged over all runs and shown as *novel detection accuracy*.

For SVM, we use libsvm [17] package, and apply C-Support Vector Classification (C-SVC) with the radial basis function using “gamma” = 0.2 and “C”=1. We use our

own C++ implementation of NB. We use the WEKA [2] implementation of J48, with pruning applied. We extract rules from the decision trees generated using J48.

6 Results

We discuss the results in three separate subsections. In section 6.1 we discuss the results found from the original data. In section 6.2, we discuss the results found from the reduced dimensional data using PCA. In section 6.3, we discuss the results obtained using J48.

6.1 Results from the Original Dataset

The results are shown in table 1. Table 1 reports the accuracy of the cross validation and novel detection for each dataset. The cross validation accuracy is shown under the column '*Acc'* and the accuracy of detecting novel worms is shown under the column '*novel detection acc'*. Each worm at the row heading is the novel worm for that dataset. In table 1, we report accuracy and novel detection accuracy for each of the six worm types. From the results reported in table 1, we see that SVM observes the best accuracy among all classifiers. The best accuracy observed by SVM is 99.77%, on the sobig.f dataset, while the worst accuracy observed by the same is 99.58%, on the mydoom.m dataset.

Table 1. Comparison of accuracy (%) of different classifiers on the worm dataset

Worm Type	**NB**		**SVM**		**SVM+NB**	
	Acc (%)	Novel detection Acc (%)	Acc (%)	Novel detection Acc (%)	Acc (%)	Novel detection Acc (%)
Mydoom.m	99.42	21.72	99.58	30.03	99.38	21.06
sobig.f	99.11	97.01	99.77	97.01	99.27	96.52
Netsky.d	99.15	97.01	99.69	65.01	99.19	64.02
Mydoom.u	99.11	97.01	99.69	96.19	99.19	96.19
Bagle.f	99.27	97.01	99.61	98.01	99.31	95.52
Bubbleboy	99.19	0	99.65	0	99.31	0
Average	**99.21**	**68.29**	**99.67**	**64.38**	**99.28**	**62.22**

6.2 Results from the Reduced Dimensional Data (Reduced by PCA)

The following chart (Fig. 1) shows the results of applying PCA on the original data. The *X* axis represents the dimension of the reduced dimensional data, which has been varied from 5 to 90, with step 5 increments. The last point on the *X* axis is the unreduced or original dimension. Fig. 1 shows the cross validation accuracy for different dimensions. The data from the chart should be read as follows: a point (x, y) on a given line, say the line for SVM, indicates the cross validation accuracy y of

SVM, averaged over all six datasets, where each dataset has been reduced to x dimension using PCA.

Fig. 1 indicates that at lower dimensions, cross validation accuracy is lower, for each of the three classifiers. But SVM is found to have achieved its near maximum accuracy when data dimension is 30. NB and SERIES reaches within 2% of maximum accuracy at dimension 30 and onwards. All classifiers attain their maximum at the highest dimension 94, which is actually the unreduced data. So, from this observation, we may conclude that PCA is not effective on this dataset, in terms of cross validation accuracy. The reason behind this poorer performance on the reduced dimensional data is possibly the one that we have mentioned earlier in section 3.2. The reduction by PCA is not producing a lower dimensional data where dissimilar class instances are maximally dispersed and similar class instances are maximally concentrated. So, the classification accuracy is lower at lower dimensions.

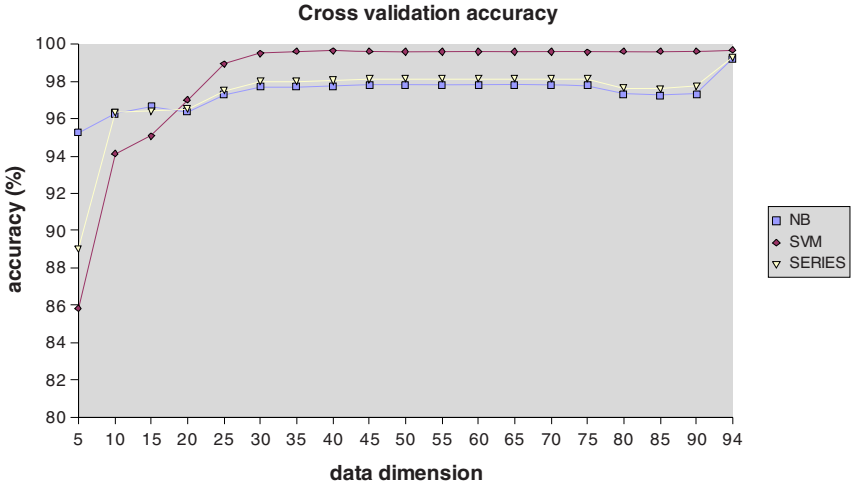

Fig. 1. Average cross validation accuracy of the three classifiers on lower dimensional data, reduced by PCA

We now present the results, at dimension 25, similar to the results presented in previous section. Table 2 compares the novel detection accuracy and the cross validation accuracy of different classifiers. We choose this particular dimension because, at this dimension all the classifiers seem to be the most balanced in all aspects: cross validation accuracy, false positive and false negative rate and novel detection accuracy. We conclude that this dimension is the optimal dimension of projection by PCA.

Results in Table 2 indicate that accuracy and novel detection accuracy of SVM are higher than NB, respectively. Also, as mentioned in previous section, we again observe that accuracy and novel detection accuracy of SVM+NB is worse than both NB and SVM. Thus, SVM is found to be the best among these three classifiers, in both unreduced and reduced dimensions.

Table 2. Comparison of accuracy among different classifiers on the PCA reduced worm dataset at dimension 25

Worm Type	NB		SVM		SVM+NB	
	Acc (%)	Novel detection Acc (%)	Acc (%)	Novel detection Acc (%)	Acc (%)	Novel Detection Acc (%)
Mydoom.m	99.08	25.0	99.46	30.02	99.15	24.7
sobig.f	97.31	97.01	99.19	97.01	97.77	97.01
Netsky.d	96.61	97.51	98.62	97.01	96.73	97.01
Mydoom.u	96.92	97.51	98.46	97.34	97.15	97.34
bagle.f	96.92	97.51	98.93	97.01	97.07	97.01
Bubbleboy	96.96	0	98.88	0	97.08	0
Average	**97.3**	**69.09**	**98.92**	**69.73**	**97.49**	**68.85**

6.3 Results Obtained from J48

Table 3 reports the accuracy, novel detection accuracy, #of selected features and tree size for different worm types. Comparing with the previous results, we find that the average novel detection accuracy of J48 is the highest (70.9%) among all the classifiers both in the original and PCA-reduced dataset. Besides, the average accuracy (99.35%) is also better than all other classifiers in the reduced dataset and very close to the best accuracy (SVM, 99.67%) in the original dataset. Surprisingly enough, on average only 4.5 features have been selected by the decision tree algorithm, which means almost 90 other features are redundant. It is interesting to see which features have been selected by the decision tree algorithm. First, we describe the rules in disjunctive normal form (DNF) that we have extracted from each of the decision trees. Each rule is expressed as a disjunction of one or more conditions. We use the symbol '$\wedge$' to denote conjunction and '$\vee$' to denote disjunction. We are able to detect the reason (explained later) behind the poor performance of all the classifiers in **Bubbleboy** dataset, where all of them have 0% novel detection accuracy.

Table 3. Accuracy (%), novel detection accuracy (%), #of selected features, and tree size as obtained by applying J48 on the original dataset

Worm Type	Acc (%)	Novel detection Acc (%)	Total features selected	Tree size (total nodes)
Mydoom.m	99.3	32.0	4	11
sobig.f	99.4	97.5	4	11
Netsky.d	99.2	99.0	4	11
Mydoom.u	99.2	97.0	6	15
bagle.f	99.4	99.5	6	15
Bubbleboy	99.6	0.5	3	7
Average	**99.35**	**70.92**	**4.5**	**11.67**

Worm rules: if any of the following rules is satisfied then it is a worm

Rule I (from Mydoom.m dataset):
[(VarWordsInBody<=457)∧(RatioAttach<=0.9)∧(MeanWordsInBody<=22.1)]
∨ [(VarWordsInBody<=457)∧(RatioAttach<=0.9)]

Rule II (from sobig.f dataset): [(RatioAttach > 0.7) ∧ (VarAttachSize <= 7799173)]
∨ [(RatioAttach>=0.7)∧(VarAttachSize>7799173)∧(NumAttachments>0)]

Rule III (from Netsky.d dataset): [(RatioAttach > 0.6) ∧ (VarAttachSize <= 10229122)]
∨ [(RatioAttach>=0.7)∧(VarAttachSize>10229122)∧(NumAttachments>0)]

Rule IV (from Mydoom.u dataset):
[(FreqEmailSentInWindow <= 0.067) ∧ (MeanWordsInBody <= 60.6)]
∨ [(FreqEmailSentInWindow<=0.067)∧(MeanWordsInBody>60.6)∧(NumAttachments>0)]

Rule V (from bagle.f dataset): [(RatioAttach > .6) ∧ (VarAttachSize <= 7799173)]
∨[(RatioAttach>.6)∧(VarAttachSize>7799173)∧NumAttachments>0)∧(AvgWordLength<45)]

Rule VI (from Bubbleboy dataset):
[(NumFromAddrInWindow>1)∧(AttachmentIsText0= 1)]

By looking at the above rules, we can easily find some important features such as:

VarWordsInBody, RatioAttach, MeanWordsInBody, NumAttachments, VarAttachSize, and so on. Using the above rules, we can also summarize general characteristics of worm. For example, it is noticeable that for most of the worms, *RatioAttach* >= 0.7, as well as NumAttachments > 0. These generalizations may lead to a generalized set of rules that would be effective against a new attack.

The rule VI above is obtained from the 'Bubbleboy' dataset. But only one of the 200 test cases satisfies this rule, so the novel detection accuracy is only 0.5%. Other classifier results also show that novel detection accuracy on the dataset is also 0%. This indicates that this worm has completely different characteristics, and cannot be detected by the generalizations obtained on other five worm types.

7 Conclusion

In this work, we explore three different data mining approaches to automatically detect email worms. The first approach is to apply either NB or SVM on the original dataset, without any feature reduction, and train a classifier. The second approach is to reduce data dimension using PCA and apply NB or SVM on the reduced data and train a classifier. The third approach is to select best features using decision tree such as J48 and obtain classification rules. Results obtained from our experiments indicate that J48 achieves the best performance. Looking at the rules extracted from the decision tree, we conclude that the feature space is actually very small, and the classifiers are quite simple. That is why the tree based selection performs better than PCA in this dataset. It might not have been the case if the feature space had been more complex. In that case, the second approach would have been more effective. The first approach would be suitable if we have only a few features in the original data. In summary, all the approaches are possible to apply, depending on the characteristic of the dataset.

In future, we would like to continue our research in detecting worms by combining feature-based approach with content-based approach to make it more robust and

efficient. Besides, rather than relying entirely on features, we are willing to focus on the statistical property of the contents of the messages for possible contamination of worms. Finally, we would also like to obtain a larger and richer collection of dataset.

References

1. Martin, S., Sewani, A., Nelson, B., Chen, K., Joseph, A.D. A Two-Layer Approach for Novel Email Worm Detection. Submitted to USENIX SRUTI (Steps on Reducing Unwanted Traffic on the Internet) 2005.
2. WEKA: Data Mining Software in Java. http://www.cs.waikato.ac.nz/~ml/weka/
3. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
4. Golbeck, J., and Hendler, J. Reputation network analysis for email filtering. In CEAS (2004).
5. Newman, M. E. J., Forrest, S., and Balthrop, J. Email networks and the spread of computer viruses. Physical Review E 66, 035101 (2002).
6. Symantec Corporation. W32.Beagle.BG. Online, 2005. http://www.sarc.com/avcenter/venc/data/w32.beagle. bg@mm.html.
7. Schultz, M., Eskin, E., and Zadok, E. MEF: Malicious email filter, a UNIX mail filter that detects malicious windows executables. In USENIX Annual Technical Conference - FREENIX Track (June 2001).
8. Singh, S., Estan, C., Varghese, G., and Savage, S. The EarlyBird System for Real-time Detection of Unknown Worms. Technical report - cs2003-0761, UCSD, 2003.
9. Kim, H.-A. and Karp, B., Autograph: Toward Automated, Distributed Worm Signature Detection, in the Proceedings of the 13th Usenix Security Symposium (Security 2004), San Diego, CA, August, 2004.
10. J. Newsome, B. Karp, and D. Song. Polygraph: Automatically Generating Signatures for Polymorphic Worms. In Proceedings of the IEEE Symposium on Security and Privacy, May 2005.
11. Honeypot. http://www.honeypots.net/.
12. Martin, S., Sewani, A., Nelson, B., Chen, K., Joseph, A.D. Analyzing Behavioral Features for Email Classification, In the Proceedings of the IEEE Second Conference on Email and Anti-Spam (CEAS 2005), July 21 & 22, 2005, Stanford University.
13. Sidiroglou, S., Ioannidis, J., Keromytis, A.D., Stolfo, S.J. An Email Worm Vaccine Architecture. Proceedings of the First International Conference on Information Security Practice and Experience (ISPEC 2005), Singapore, April 11-14, 2005: 97-108
14. Mitchell, T. Machine Learning. McGraw Hill, 1997.
15. John, G.H., Langley, P.: Estimating Continuous Distributions in Bayesian Classifiers. In the Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan KaufMann Publishers, San Mateo, (1995) 338-345.
16. Boser, B. E., Guyon, I. M. and Vapnik, V. N. A training algorithm for optimal margin classifiers. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992. ACM Press.
17. A library for Support Vector Machine: http://www.csie.ntu.edu.tw/~cjlin/libsvm/