Detecting a Malicious Executable without Prior Knowledge of Its Patterns

[a]D. Michael Cai[1], James Theiler[2], and Maya Gokhale[1]

[1]Space Data Systems Group and [2]Space and Remote Sensing Group
Los Alamos National Laboratory, Los Alamos, NM 87545

ABSTRACT

To detect malicious executables, often spread as email attachments, two types of algorithms are usually applied under instance-based statistical learning paradigms: 1) Signature-based template matching, which finds unique tell-tale characteristics of a malicious executable and thus is capable of matching those with known signatures; 2) Two-class supervised learning, which determines a set of features that allow benign and malicious patterns to occupy a disjoint regions in a feature vector space and thus probabilistically identifies malicious executables with the similar features. Nevertheless, given the huge potential variety of malicious executables, we cannot be confident that existing training sets adequately represent the class as a whole. In this study, we investigated the use of byte sequence frequencies to profile only benign data. The malicious executables are identified as outliers or anomalies that significantly deviate from the normal profile. A multivariate Gaussian likelihood model, fit with a Principal Component Analysis (PCA), was compared with a one-class Support Vector Machine (SVM) model for characterizing the benign executables. We found that the Gaussian model substantially outperformed the one-class SVM in its ability to distinguish malicious from benign files. Complementing to the capabilities in reliably detecting those malicious files with known or similar features using two aforementioned methods, the one-class unsupervised approach may provide another layer of safeguard in identifying those novel computer viruses.

Keywords: Mining high-dimensional data, One-class Support Vector Machine, Principal Component Analysis, Malicious Executable File, Email Filtering

1. INTRODUCTION

While originally conceived as a convenient tool for text messages, email has since evolved into the backbone of the Internet, and has become the primary medium not only for communicating ideas, opinions, and appointments, but also for unauthorized accesses

[a] Email: {dmc, jt, maya}@lanl.gov

and malicious attacks. For instance, a malicious executable program attached to an apparently benign email can easily be sent to thousands of recipients.

Building a reliable computer security system to detect malicious codes resembles, in many aspects, a natural immune system that protects animals from dangerous foreign pathogen including bacteria, viruses, and toxins. The immune system comprises cells and molecules and possesses an elegant self-defense mechanism distinguishing "self" from dangerous "other" and eliminating other. To detect foreign pathogens, immune systems remember previous infections and mount a more aggressive response against those seen before, a so-called a secondary response. In the case of a novel infection, the immune system initiates a primary response, evolving new detectors specialized for the new infection. The process is not so quick as a secondary response but provides an essential capability of novelty or anomaly detection that is lacking in many computer security systems [10].

To detect malicious executables, two types of algorithms have been applied under instance-based statistical learning paradigms: 1) Signature-based template matching, which finds unique tell-tale characteristics of a malicious executable and thus is capable of matching those with known signatures [5][7][8][13][14]; 2) Two-class supervised learning, which determines a set of features that allow benign and malicious patterns to occupy a disjoint regions in a feature vector space and thus probabilistically identifies malicious executables with the similar features [3][12] [18][19][23]. Nevertheless, given the huge potential variety of malicious executables, we cannot be confident that existing training sets adequately represent the class as a whole.

In this study, we investigated the use of byte sequence frequencies to profile only benign data [3][18]. The rationale for choosing only byte sequences as candidate features is that those byte patterns are the most accessible and reliable information that represents the machine code in an executable. Secondly, using embedded text strings as features, such as head information, program names, authors' names, or comments, is not robust since they can be easily changed. Some malicious executables intentionally camouflage these signatures by randomly generating these fields to deceive virus scanners [12][18]. The malicious executables are identified as outliers or anomalies that significantly deviate from the normal profile. This unsupervised learning approach does not require predefining the malicious patterns to be classified. Instead, it characterizes only patterns of known benign files, which bears a resemblance to an immune system distinguishing "self" from dangerous "other". A multivariate Gaussian likelihood model, fit with a Principal Component Analysis (PCA) [11], was compared with a one-class Support Vector Machine (SVM) model [17][20] for characterizing the benign executables.

2. METHODS

2.1 Data Description and Preparation

Our experimental data were downloaded from the Intrusion Detection System site at Columbia University: http://www.cs.columbia.edu/ids/mef/. It consists of total 4754 files

with 1074 clean binary files and 3680 malicious ones. There are no duplicate programs in the data set, and each file is labeled either malicious or benign. A more detailed description of the dataset can be found in [18][19]. The downloaded files have been already transformed from their binary form into hexadecimal text format. A byte sequence pattern consists of n sequential bytes; for $n = 1$, the pattern consists of a single byte (*e.g.*, **07, eb, 0a, 56**, *etc.*), for $n = 2$, two bytes (*e.g.*, **07eb**, **eb0a**, **0a56**, *etc.*), so on. Based on our previous results [3], there seems no advantage to use multiple byte sequences as features, and so only the frequencies of single byte patterns are accounted for potential features.

2.2 Classification Algorithms

2.2.1 Principal Component Analysis (PCA)

Principal component analysis [11] is frequently used in order to uncover significant dimensions underlying a large set of data. Let $\boldsymbol{x_1, x_2, \ldots, x_n}$ denote n samples of dimension d. These data points can be imagined to scatter with an orthogonal system of d axes. One might wish to rotate the d axes by applying an orthogonal transformation **P** in such a way that the new coordinates describe the d variables in a simpler manner. For instance, if the data points form a d-dimensional ellipsoid, we would wish to rotate the principal axes so that they lie along the directions with the maximum data variance. Principal component analysis chooses the m ($m \leq d$) largest eigenvectors of the $d \times d$ covariance matrix of the n d-dimensional patterns. The linear transformation is defined as:

$$\mathbf{Y} = \mathbf{X} * \mathbf{P} \qquad (1)$$

Where **X** is the given $n \times d$ pattern matrix, **Y** is the derived $n \times m$ pattern matrix (termed "scores"), and **P** is the $d \times m$ matrix of linear transformation whose columns are the eigenvectors corresponding to the m largest eigenvalues. The purpose of principal component analysis is then to uncover the m "common" principal components that "explain" the data and account for the maximum possible variance. The remaining d-m "unique" components are viewed as accounting for nonessential variance reflecting the individual differences or uniqueness of the n sample points.

We employed MATLAB statistical toolbox to perform principal component analysis on the training set [11][15]. To quantify the disparity of the data, Hotelling's T^2, a statistical measure of the multivariate distance of each observation from the center of the data set, was computed. The same transformations were applied to calculate T^2 statistics on the testing sets with both benign and malicious files.

2.2.2 One-class Support Vector Machine (SVM)

Support Vector Machines (SVMs) were originally introduced by Vapnik and his colleagues for solving the two-class pattern recognition problem [6][24]. The idea that

uses kernels to compute inner products in feature space was recently extended to the domain of unsupervised learning [17][20]. The one-class SVM used in this study is based on the work Schölkopf and his colleagues [17].

One-class SVMs produce a model of the dataset even though the data samples are all from a single class. The goal is to find a "smallest" set which encloses a specified fraction (usually something like 95% or 99%) of the underlying probability distribution for the one class. Since that underlying distribution is not available, it must be estimated from the finite sample. A natural interpretation of "smallest" is minimum volume, but in the one-class SVM, particularly the kernelized version, it is the "smoothest" description (i.e., smallest coefficients) that is desired. For kernel functions (such as the Gaussian radial basis function kernel) which depend only on the distance |x-y|, this corresponds to the smallest volume sphere in kernel space. The form of the solution is a function

$$f(\boldsymbol{x}) = \sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \qquad (2)$$

where k(x,y) is the kernel function, and the nonzero values of the coefficients α_i correspond to the support vectors. The sign of f(x) determines whether or not x is predicted to be in the class.

The software package we used to generate SVM classifiers is called *LIBSVM* which was developed by Chih-Chung Chang and Chih-Jen Lin [4]. The software is available for downloading at http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

The kernel function used in this study is radial basis function:

$$K(u,v) = \exp(-\gamma \mid u - v \mid^2) \qquad (3)$$

Its width γ was found to affect prediction accuracy of one-class SVMs and the results were reported in the following sections.

2.3 Performance Measures

The effectiveness is evaluated using the performance of different classifiers defined in last section. To enable direct comparison, the model performance uses a similar measure to that used by [18][19].

1. True Positives (TP), the number of malicious executables correctly classified as malicious;
2. True Negatives (TN), the number of benign programs correctly classified as benign;
3. False Positives (FP), the number of benign programs falsely classified as malicious,

4. False Negatives (FN), the number of malicious executables falsely classified as benign.

$$\text{Detection Rate (DTR)} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN + FP} \quad (5)$$

$$\text{Overall Accuracy (OA)} = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

2.4 Experimental Setting

The results reported in our experiments are based on five-fold cross-validation. The malicious and benign files were randomly partitioned into five sets. One of the five (comprising 20% of the data samples) was held out as a test set and the remaining four (80% of the data) were concatenated into a training set. Note that the trainings set only consisted of benign files, but the validation sets were mixed with both benign and malicious executables. This was done five times, once for each choice of training set, and the performance was averaged over these five trials.

3. RESULTS

To illustrate differences between benign and malicious executables after their frequencies of 256 single byte patterns have been transformed by an eigenvector matrix **P**, in Figure 1, we plot their exemplar distributions on scatter diagrams below

On the upper panel, the scores of 1st and 2nd principal components of benign files (blue, or "dark" if seen in gray scale) and malicious files (magenta, or "light") are overlaid. Most benign data points sampled from the training set are intermingled with the malicious ones, which indicates the first and second components could well explain the maximum variance of feature variables but may not be the best features to distinguish two classes. However, on the lower panel, the comparison was made on the scores of 254th and 255th components. As expected, the data from both classes are present in smaller dynamic ranges. The benign data points are tightly clustered near the origin, and the majority of malicious data points also forms a cluster but with a larger spread. And the middle panel illustrates a pattern qualitatively in between. Note that the eigenvector matrix **P** was calculated on the pattern matrix **X** of the training set, and the same matrix **P** then projected both benign and malicious data points from the test set to the principal axes. The patterns of benign and malicious files on the principal components suggest

excluding some top principal components as features since two classes are not well separated. Also, the magnitude of eigenvalues vary from 5 to 6 orders and the projection

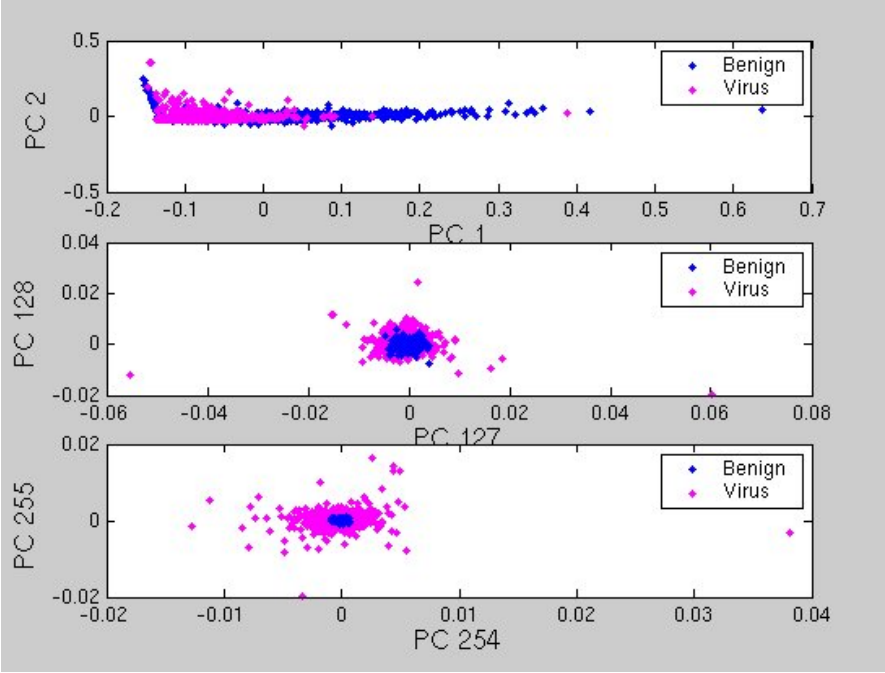

Figure 1: Comparison of principal components between benign (the training set) and malicious files.

to those components corresponding to extremely small eigenvalues might be more influenced by round-off errors and by the larger expected relative differences between the training and testing at that scale. Therefore some of principal components in the tail were dropped as well.

In our next table, we show the classification made by Hotelling's T^2 measure calculated from a feature set containing 88^{th}—158^{th} principal components. Principal components were computed for the training data and these were used to define the T^2 formula that was then applied to the test data. We experimented with different thresholds of T^2, which yielded different prediction accuracies on in-sample data.

False Positive Rate (%)	Detection Rate (%)
1.58	42.88
1.86	59.57
2.70	71.66
3.63	86.79
4.93	91.60
6.61	94.19
8.10	95.95
11.64	98.04

Table 1: Classification made by Hotelling's T^2 measure. The classifiers were trained only on benign but validated on the testing sets containing both benign and malicious files.

The classifier gives reasonable performance in terms of prediction accuracy. For instance, at 5% false positive rate, it detected over 92% of the unseen malicious files.

In the following experiments, we show the results made by one-class SVM classifiers. The classifiers were trained on benign data. One-class SVMs take the input data either in their original format (*un-scaled*) or scaled format in which each component of the data is individually scaled linearly between –1 to 1 using a utility "svm-scale" included in LIBSVM package. In Table 2, the results show parameter ν, which is an upper bound on the fraction of outliers and lower bound on the fraction of SVs (Schölkopf, 2001), correlates with false positive rate monotonically. The separation between benign and malicious class is very modest. The one-class SVM trained on scaled data has slightly better prediction accuracy, especially when the false positive rate is low.

γ= *1/256*	**Un-scaled Data**		**Scaled Data**	
ν	**False Positive Rate (%)**	**Detection Rate (%)**	**False Positive Rate (%)**	**Detection Rate (%)**
0.01	1.68	1.22	1.49	2.94
0.02	2.23	1.32	2.42	5.43
0.05	4.93	3.27	5.21	15.27
0.10	9.87	8.45	9.87	33.22
0.20	19.92	24.86	19.83	65.56
0.30	29.98	43.43	30.26	83.99
0.50	50.84	75.73	50.83	94.22
0.70	71.23	93.43	70.20	98.77
0.90	89.11	99.67	89.94	99.82

Table 2: Performance of one-class SVM on un-scaled and scaled data. RBF kernels were used. γ=1/256.

In Table 3, when parameter γ (width of Gaussian RBF) was increased from 1/256 to 1, the impact to the one-class SVM trained on the un-scaled data was almost undetectable. Nevertheless, for the SVM trained on scaled data, both false positive rate and detection rate increased significantly. The trend seems towards an improvement of overall prediction accuracy.

γ=1	Un-scaled Data		Scaled Data	
ν	False Positive Rate (%)	Detection Rate (%)	False Positive Rate (%)	Detection Rate (%)
0.01	1.30	1.20	23.84	91.35
0.02	2.05	1.48	23.93	91.36
0.05	5.59	3.83	24.49	91.36
0.10	9.87	9.31	24.58	91.37
0.20	20.30	26.32	26.72	91.80
0.30	30.26	45.17	33.42	94.79
0.50	50.74	75.98	50.47	97.78
0.70	70.11	93.32	70.39	99.64
0.90	90.04	99.78	90.13	99.97

Table 3 Performance of one-class SVM on un-scaled and scaled data. RBF kernels were used. γ=1.

In Figure 2, we illustrate classification performance with a plot of the ROC (Receiver Operating Characteristic) curve that shows the trade-off between detection rate (y-axis) and false positive rate (x-axis).

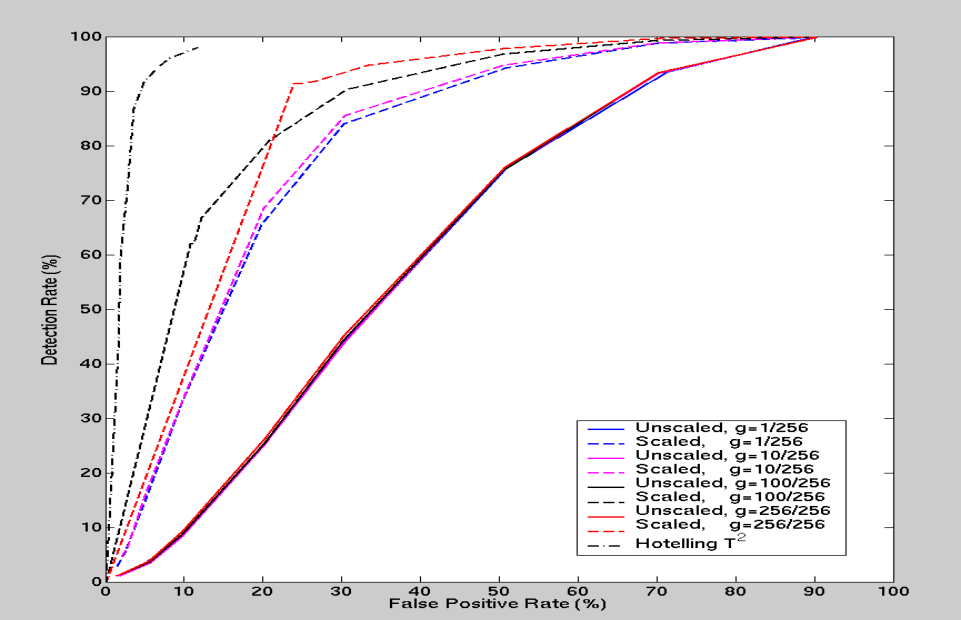

Figure 2: Performance comparison of one-class SVMs as a function of parameter γ with un-scaled and scaled input data. The results made by T^2 measure are displayed on the curve at the most left plot.

Ideally, we want a high detection rate (to detect most of the malicious files) and a low false alarm rate (to avoid mistakenly classifying benign files as malicious). The parameter γ *was* chosen at 1/256, 10/256, 100/256, and 1, respectively with un-scaled data (solid lines) and scaled data (dashed lines). When γ is increased from 1/256 to 1, for the un-scaled training data, the performance of one-class SVMs remains unchanged. On the flip side, the performance in terms of prediction accuracy has been improved when one-class SVMs were trained on the scaled data. To give a direct performance comparison between the PCA and one-class SVMs, the ROC of Hotelling's T^2 was also plotted in this figure (dash-dotted line). It is clear that the prediction accuracy performed by T^2 statistic [is] better than all the one-class SVM variants.

4. DISCUSSION

In a traditional two-class supervised learning paradigm, we have exemplar data from both classes, and the decision boundary is supported from both sides of samples to maximize the separation between them. In one-class classification, only one class of training data is available. It is very hard to determine how tight the decision boundary should be to best characterize the known class. It is even more difficult to determine which features should be selected to yield the best separation of the known class and other classes.

There are several ways to study one-class classification problem. For example, use of artificial outlier data [16], weighted outputs [2], directly estimate the probability density of the known class [1][9] and recent progress on estimation of minimal volume representing the known class [17][20]. For a more complete review on the subject of one-class classification, see [21].

In this study, motivated by the concept of computer immunology [10], we took an unsupervised approach to detect malicious executables, a real threat to network security and user privacy. A multivariate Gaussian likelihood model, fit with Principal Component Analysis (PCA), was compared with a one-class Support Vector Machine (SVM) model for characterizing the benign executables. Comparing to the supervised learning approaches [3][18], in which both benign and malicious data are required in training set, the one-class unsupervised approach tries to optimize the decision boundary from one side. It could play a complementary role to detect those novel malicious files that are likely missed using both signature-based template matching and supervised learning-based detection.

Using principal component analysis, the benign and malicious data points were projected to principal axes, as determined from a training set of only benign files. As our results showed (Figure 1), the data of two classes were intermingled in some principal components, and clustered differently in some other principal axes. The hope was to find a set of selected principal components as features, and a proper distance metric (Hotelling's T^2 statistic in this case) to separate two classes.

Our results show the PCA approach generates reasonable performance in terms of prediction accuracy. For example, at 5% false positive rate, it detected 92% of unseen

malicious executables correctly. Also, the performance between in-sample training and out-of-sample validation is consistent.

On the other side, an elongated shape of input data seems to make one-class SVM perform poorly. For instance, at about 93% detection rate, one-class SVM has as high as 70% false positive rate on un-scaled data. Scaling input data, and increasing the value of gamma i.e. deceasing the width of radial basis kernels improved the performance of one-class SVM. Nevertheless, its performance does not outperform the PCA approach.

This data scaling approach has the similar performance improvement reported by Tax and Juszczak [22] using kernel whitening for data description. Their data is mapped onto the principal components in a kernel space and then rescaled by the corresponding eigenvalues. Whitening the data from elongated to spherical clusters, or scaling data to be more uniformly distributed could improve performance of one-class SVM.

5. ACKOWLEDGEMENTS

We are grateful to Dr. Stolfo at the Columbia University for letting us use their data. We also appreciate Drs. Chih-Jen Lin and Chih-Chung Chang for providing the LIBSVM package. The work was supported by the LDRD program and DAPS program at Los Alamos National Laboratory.

6. REFERENCES

[1] V. Barnett and T. Lewis, Outliers in Statistical Data, (John Wiley & Sons Ltd, 1978).

[2] C. Bishop, Neural Networks for Pattern Recognition, (Oxford University Press, Oxford, 1995).

[3] D. M. Cai, M. Gokhale and J. Theiler, Comparison of Feature Selection and Classification Algorithms in Identifying Malicious Executables, Submitted to Data and Knowledge Engineering 2003.

[4] C. C Chang and C.J. Lin, LIBSVM: a Library for Support Vector Machines, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. 2001.

[5] F. Cohen, A Cryptographic Checksum for Integrity Protection, Computers and Security 6 (1987) 505-510.

[6] C. Cortes and V. Vapnik, Support Vector Network, Machine Learning 20 (1995) 273-297.

[7] R. Crawford, R. Lo, J. Crossley, G. Fink, P. Kerchen, W. Ho, K. Levitt, R. Olsson, and M. Archer, A Testbed for Malicious Code Detection: A Synthesis of Static and Dynamic Analysis Techniques, Proceedings of the Department of Energy Computer Security Group Conference, 17, pp. 1-23, 1991.

[8] S. Crocker and M. M. Pozzo, A Proposal for a Verification to Malicious Code Detection, Proceedings of IEEE Computer Soc. Symposium on Security and Privacy, pp. 319-324, 1989.

[9] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in Unlabeled Data. On Application of Data Mining in Computer Security, Edited by D. Barbará and S. Jajodia, (Kluwer Academic Publisher, 2002).

[10] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer Immunology, Communications of the ACM, pp. 88-9, 1997.
[11] J. E. Jackson, A User's Guide to Principal Components, (John Wiley and Sons, 1991).

[12] J. O. Kephart and W. C. Arnold, Automatic Extraction of Computer Virus Signatures, 4th Virus Bulletin International Conference, pp. 178-184, 1994.

[13] R. W. Lo, P. Kerchen, R. Crawford, W. Ho, J. Crossley, G. Fink, K. Levitt, R. Olsson, and M. Archer, Towards for Malicious Code Detection, IEEE Computer Society International Conference, pp. 160-166, 1991.

[14] R. W. Lo, Karl N. Levvit, and Ronald A. Olsson, MCF: A Malicious Code Filter, Computers & Security, 14, pp. 541-566, 1995.

[15] MathWorks, Statistical Toolbox for User's Guide, MathWorks, 2001.

[16] S. Roberts, L. Tarassenko, J. Pardey, and D. Siegwart, A validation index for artificial neural network, In Proceedings of Int. Conference on Neural Networks and Expert System in Medicine and Healthcare, pp. 23-30, 1994.

[17] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, Estimating the support of high-dimensional distribution, Neural Computation (13), pp. 1443-1471, 2001.

[18] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, Data Mining Methods for Detection of New Malicious Executables, Proceedings of IEEE Symposium on Security and Privacy. Oakland, pp.38-49, 2001.

[19] M. G. Schultz, E. Eskin, and S. J. Stolfo, Malicious Email Filter - A UNIX Mail Filter that Detects Malicious Windows Executables, Proceedings of USENIX Annual Technical Conference - FREENIX Track. Boston, MA: June, 2001.

[20] D. Tax and R. Duin, Data domain description by support vectors, in Proc. ESANN, M. Vefieysen, Ed., Brussels, pp. 251 -256, D. Facto Press, 1999.

[21] D. M. J. Tax, One-class classification: concept-learning in the absence of counter-examples, Doctorate Dissertation, TU Delft, 2001.

[22] D.M.J. Tax and P. Juszczak. Kernel whitening for data description, International Workshop on Pattern Recognition with Support Vector Machines 2002, Niagara Falls, Canada.

[23] G. Tesauro, J. O. Kephart, and G. B. Sorkin, Neural Networks for Computer Virus Recognition, IEEE Expert, 11 (1996) 5-6.

[24] V. Vapnik, Statistical Learning Theory, (New York, John Wiley & Sons, Inc., 1998).