Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems

Roberto Perdisci[†,‡], Guofei Gu[‡], Wenke Lee[‡]

[‡]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
[†]DIEE, University of Cagliari, 09031 Cagliari, ITALY
{rperdisc,guofei,wenke}@cc.gatech.edu

Abstract

Unsupervised or unlabeled learning approaches for network anomaly detection have been recently proposed. In particular, recent work on unlabeled anomaly detection focused on high speed classification based on simple payload statistics. For example, PAYL, an anomaly IDS, measures the occurrence frequency in the payload of n-grams. A simple model of normal traffic is then constructed according to this description of the packets' content.

It has been demonstrated that anomaly detectors based on payload statistics can be "evaded" by mimicry attacks using byte substitution and padding techniques. In this paper we propose a new approach to construct high speed payload-based anomaly IDS intended to be accurate and hard to evade. We propose a new technique to extract the features from the payload. We use a feature clustering algorithm originally proposed for text classification problems to reduce the dimensionality of the feature space. Accuracy and hardness of evasion are obtained by constructing our anomaly-based IDS using an ensemble of one-class SVM classifiers that work on different feature spaces.

1 Introduction

Intrusion Detection Systems (IDS) are valuable tools for the defense-in-depth of computer networks. Network IDS look for known or potential malicious activities in network traffic and raise an alarm whenever a suspicious activity is detected. Two main approaches to intrusion detection are used, namely *misuse* and *anomaly* detection [19]. Misuse detectors are based on a description of known malicious activities. This description is often modeled as a set of rules referred to as *attack signatures*. Activities that match an attack signature are classified as malicious. Anomaly detectors are based on a description of *normal* or *benign* activities. A distance between the description of normal events and new network activities is measured. As malicious activities are expected to be different from normal activities, a suitable distance measure allows anomaly-based IDS to detect attack traffic. Anomaly-based detection systems usually produce a relatively higher number of false positives, compared to the misuse-based or *signature-based* detection systems. However, anomaly detectors are able to detect *zero-day* (i.e., never-before-seen) attacks, whereas signature-based systems are not.

Unsupervised or *unlabeled* learning approaches for network anomaly detection have been recently proposed [20, 10]. These methods aim to work on datasets of traffic extracted from real networks without the necessity of a labeling process. *Unlabeled* anomaly detection systems are based on the reasonable assumption that the percentage of attack patterns in the extracted traffic traces is usually much lower than the percentage of normal patterns [20]. Furthermore, it is possible to use signature-based IDS in order to filter the extracted traffic by removing the known attacks, thus further reducing the number of attack patterns possibly present in the dataset. Another assumption is that the attack patterns are supposed to be distinguishable from the normal patterns in a suitable feature space. The term "unlabeled anomaly detection" used in the intrusion detection field actually refears to what in machine learning is more often called "novelty detection", "outlier detection" or "one-class classification". One-class classification algorithms pursue concept learning in absence of counter examples [23]. The objective of one-class classification is to find a decision surface around the *target objects*, (i.e., the normal traffic, in case of network anomaly detection) so that patterns that lie inside this decision surface are classified as *targets* (i.e., normal traffic), whereas patterns that lie outside are classified as *outliers* (i.e., anomalous traffic).

Recent work on unlabeled anomaly detection focused on *high speed* classification based on simple *payload*[*] statistics [15, 18, 28, 29]. For example, PAYL [28, 29] extracts 256 features from the payload. Each feature represents the occurrence frequency in the payload of one of the 256 possible byte values. A simple model of normal traffic is then constructed by computing the average and standard deviation of each feature. A payload is considered anomalous if a *simplified Mahalanobis distance* between the payload under test and the model of normal traffic exceeds a predetermined threshold. Wang et al. [28] also proposed a more generic n-gram[†] version of PAYL. In this case the payload is described by a pattern vector in a 256^n-dimensional feature space. The n-grams extract byte sequence information from the payload, which helps in constructing a more precise model of the normal traffic compared to the simple byte frequency-based model. The extraction of n-gram statistics from the payload can be performed efficiently and the IDS can be used to monitor high speed links in real time. However, given the exponentially growing number of extracted features, the higher n the more difficult it may be to construct an accurate model because of the curse of dimensionality and possible computational complexity problems. Other anomaly detection systems based on more complex features have been proposed [25, 5]. These anomaly detectors involve the extraction of syntax and semantic information from the payload, which is usually a computationally expensive task. Therefore, it may not be possible to use this approaches in order to analyze network traffic on high speed links in real time.

It has been demonstrated that many anomaly detection systems can be "evaded" by *mimicry* attacks [27, 14, 6, 12]. A mimicry attack is an attack against a network or system vulnerability that is carefully crafted so that the attack pattern, i.e., the representation of the attack used during the classification process, lies inside the decision surface that separates the normal patterns from the anomalous ones (i.e., the *outliers*). A successful mimicry attack is able to exploit the targeted vulnerability while causing the anomaly IDS to produce a false negative (i.e., no alarm is raised). In [12], Fogla et al. showed how to construct a mimicry attack, called *polymorphic blending attack*, that can evade 1-gram (i.e., the *single-byte frequency* version) and 2-gram PAYL. Using byte substitution and padding techniques, the polymorphic blending attack encodes the attack payload so that the obtained *transformed* attack is classified as normal by PAYL, while still being able to exploit the targeted vulnerability.

In order to make it harder for the attacker to evade the IDS, a more comprehensive model of the normal traffic is neeeded. Furthermore, the modeling technique needs to be also practical and efficient. We address these challenges using an ensemble of classifiers. Classifier ensembles, often referred to as Multiple Classifier Systems (MCS), have been proved to achieve better accuracy in many applications, compared to the best single classifier in the ensemble. A number of security related applications of MCS have been proposed in the literature. For example, MCS are used in multimodal biometrics for hardening person identification [3], and in misuse-based IDS [13] to improve the detection accuracy. To the best of our knowledge, no work has been presented so far that explicitly addresses the problem of increasing the *hardness of evasion* of anomaly-based IDS using multiple classifier systems. In this paper we propose a new approach to construct a *high speed* payload-based anomaly IDS by combining multiple one-class SVM classifiers. Our approach is intended to improve both the detection accuracy and the hardness of evasion of high speed anomaly detectors.

MCS attain accuracy improvements when the combined classifiers are "diverse", i.e., they make different errors on new patterns [8]. A way to induce diversity is to combine classifiers that are based on descriptions of the patterns in different feature spaces [16]. We propose a new technique to extract the features from the payload that is similar to the 2-gram technique. Instead of measuring the frequency of the pairs of consecutive bytes, we propose to measure the features by using a sliding window that "covers" two bytes which are ν positions apart from each other in the payload. We refere to this pairs of bytes as 2_ν-grams. The proposed featrue extraction process do not add any complexity with respect to the traditional 2-gram technique and can be performed efficiently. We also show that the proposed technique allows us to "summarize" the occurrence frequency of n-grams, with $n > 2$, thus capturing byte sequence information while limiting the dimensionality of the feature space. By varying the parameter ν, we construct a representation of the payload in different feature spaces. Then we use a feature clustering algorithm originally proposed in [7] for text classification problems to reduce the dimensionality of the different feature spaces where the payload is represented. Detection accuracy and hardness of evasion are obtained by constructing our anomaly-based IDS using a combination of multiple one-class SVM classifiers that work on these different feature spaces. Using multiple classifiers forces the attacker to devise a mimicry attack that evades multiple models of normal traffic at the same time, which is intuitively harder than evading just one model. We compare our payload-based anomaly IDS to the original implementation of 1-gram PAYL [28] by Columbia University, to an implementation of 2-gram PAYL, and to an IDS constructed by combining multiple one-class classifiers based on the *simplified Mahalanobis distance* used by PAYL.

Our work is organized as follows. Section 2 presents one-class classification and the learning algorithms we used

[*] The *payload* is the data portion of a network packet.

[†] Here an n-gram represents n consecutive bytes in the payload

to perform our experiments. We discuss how the features are extracted from the payload and the algorithm we used to perform feature reduction in Section 3. Section 4 presents the experimental results and then we briefly conclude in Section 5.

2 One-Class Classifiers

One-class classification techniques are particularly useful in case of *two-class* learning problems whereby one of the classes, referred to as *target class*, is well-sampled, whereas the other one, referred to as *outlier class*, is severely undersampled. The low number of examples from the outlier class may be motivated by the fact that it is too difficult or expensive to obtain a significant number of training patterns of this class [23]. The goal of one-class classification is to construct a decision surface around the examples from the target class in order to distinguish between *target objects* and all the other possible objects, i.e., the *outliers* [23]. A *rejection rate* is usually chosen during training so that a certain percentage of training patterns lies outside the constructed decision surface in order to take into account the possible presence of noise (i.e., unlabeled outliers) in the training set and to obtain a more precise description of the target class [23]. In the case when the training set contains only "pure" target patterns, this rejection rate can be interpreted as a *tolerable* false positive rate.

In the following, we present two different one-class classification algorithms that we used to perform our experiments, namely a classifier inspired by the Support Vector Machine (SVM) [26], and a classifier based on the Mahalanobis distance [9]. As we discuss in Section 3, there is an analogy between anomaly detection based on n-gram statistics and text classification problems. We chose the one-class SVM classifier because SVM have been shown to achieve good performances in text classification problems [22, 17]. We also describe the Mahalanobis distance based classification algorithm because it is the same classification algorithm used by PAYL [28], a recently proposed anomaly detector based on n-gram statistics.

2.1 One-Class SVM

A one-class classifier inspired by the SVM classifier [26] was proposed by Schölkopf et al. in [21]. The one-class classification problem is formulated to find a hyperplane that separates a desired fraction of the training patterns from the origin of the feature space $\mathbb{F}$. This hyperplane cannot be always found in the original feature space, thus a mapping function $\Phi : \mathbb{F} \rightarrow \mathbb{F}'$, from $\mathbb{F}$ to a kernel space $\mathbb{F}'$, is used. In particular, it can be proven that when the gaussian kernel

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = exp\left(-\gamma||\mathbf{x} - \mathbf{y}||^2\right) \quad (1)$$

is used it is always possible to find a hyperplane that solves the separation problem. The problem is formulated as follows:

$$\min_{\mathbf{w},\xi,\rho} \left(\frac{1}{2} \left\|\mathbf{w}\right\|^2 - \rho + \frac{1}{mC} \sum_i \xi_i \right) \quad (2)$$
$$\mathbf{w} \cdot \phi\left(\mathbf{x}_i\right) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1,..,m$$

where $\mathbf{w}$ is a vector orthogonal to the hyperplane, C represents the fraction of training patterns that are allowed to be rejected (i.e., that are not separated from the origin by the hyperplain), $\mathbf{x}_i$ is the i-th training pattern, m is the total number of training patterns, $\xi = [\xi_1, .., \xi_m]$ is a vector of slack variables used to "penalize" the rejected patterns, ρ represents the margin, i.e., the distance of the hyperplane from the origin.

The solution of (2) brings to the decision function, for a generic test pattern $\mathbf{z}$, formulated as

$$f_{svc}(\mathbf{z}) = I\left(\sum_i \alpha_i K\left(\mathbf{x}_i, \mathbf{z}\right) \geq \rho\right), \quad \sum_{i=1}^m \alpha_i = 1 \quad (3)$$

where I is the indicator function, whereby $I(x) = 1$ if x is true, otherwise $I(x) = 0$. The coefficients α_i and the threshold ρ are provided by the solution of (2). According to (3), a pattern $\mathbf{z}$ is either rejected if $f_{svc}(\mathbf{z}) = 0$, or accepted as target object if $f_{svc}(\mathbf{z}) = 1$. It is worth noting that most of the coefficients α_i are usually equal to zero, therefore $f_{svc}(\mathbf{z})$ can be efficiently computed. The training patterns $\mathbf{x}_i$ for which $\alpha_i \neq 0$ are referred to as *support vectors*.

2.2 Mahalanobis Distance-based Classifier

Given a training dataset $D = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_m\}$, the average ϕ_i and standard deviation σ_i are computed for each feature x_{k_i}, $i = 1, .., l$, of a pattern $\mathbf{x}_k \in D$. We call $M(\phi, \sigma)$ the model of normal traffic, where $\phi = [\phi_1, \phi_2, .., \phi_l]$ and $\sigma = [\sigma_1, \sigma_2, .., \sigma_l]$. Assuming the features to be uncorrelated, a *simplified Mahalanobis distance*[‡] [28] $\Delta(\mathbf{z}, M(\phi, \sigma))$ between a generic pattern $\mathbf{z} = [z_1, z_2, .., z_l]$ and the model $M(\phi, \sigma)$ can be computed as

$$\Delta(\mathbf{z}, M(\phi, \sigma)) = \sum_{i=1}^{l} \frac{|z_i - \phi_i|}{\sigma_i + \alpha} \quad (4)$$

where α is a constant *smoothing factor* introduced in order to avoid division by zero. Given a threshold θ, the decision rule for the classifier can be written as

$$\Delta(\mathbf{z}, M(\phi, \sigma)) > \theta \ \Rightarrow\ \mathbf{z} \text{ is an outlier} \quad (5)$$

[‡] The *simplified Mahalanobis distance* do not involve square operations, which would slow down the computation of the distance.

The threshold θ can be computed during training so that a chosen rejection rate r of patterns in D is left outside the decision surface, i.e., the classifier produces a false positive rate r on the training dataset D, if we assume D contains only examples extracted from the target class.

3 Payload Classification

3.1 Feature Extraction

The detection model used by PAYL [28] is based on the frequency distribution of the n-grams (i.e., the sequences of n consecutive bytes) in the payload. The occurrence frequency of the n-grams is measured by using a sliding window of length n. The window slides over the payload with a step equal to one byte and counts the occurrence frequency in the payload of the 256^n possible n-grams. Therefore, in this case the payload is represented by a pattern vector in a 256^n-dimensional feature space. It is easy to see that the higher n, the larger the amount of structural infomation extracted from the payload. However, using $n = 2$ we already obtain 65,536 features. Larger values of n are impractical given the exponentially growing dimensionality of the feature space and the curse of dimensionality problem [9]. On the other hand, by measuring the occurrence frequency of pairs of bytes that are ν positions (i.e., ν bytes) apart from each other in the payload, it is still possible to extract some information related to the n-grams, with $n > 2$. We call such pairs of bytes 2_ν*-grams*. In practice, the occurrence frequency of the 2_ν-grams can be measured by using a $(\nu+2)$ long sliding window with a "gap" between the first and last byte.

Consider a payload $B = [b_1, b_2, .., b_l]$, where b_i is the byte value at position i. The occurrence frequency in the payload B of an n-gram $\beta = [\beta_1, \beta_2, .., \beta_n]$, with $n < l$, is computed as

$$f(\beta|B) = \frac{\text{\# of occurrences of } \beta \text{ in } B}{l - n + 1} \tag{6}$$

where the number of occurrences of β in B is measured by using the sliding window technique, and $(l-n+1)$ is the total number of times the window can "slide" over B. $f(\beta|B)$ can be interpreted as an estimate of the probability $p(\beta|B)$ of finding the n-gram β (i.e., the sequence of consecutive bytes $[\beta_1, \beta_2, .., \beta_n]$) in B. Accordingly, the probability of finding a 2_ν-gram $\{\beta_1, \beta_{\nu+2}\}$ can be written as

$$p(\{\beta_1, \beta_{\nu+2}\}|B) = \sum_{\beta_2,..,\beta_{\nu+1}} p([\beta_1, \beta_2, .., \beta_{\nu+1}, \beta_{\nu+2}]|B) \tag{7}$$

It is worth noting that for $\nu = 0$ the 2_ν-gram technique reduces to the "standard" 2-gram technique. When $\nu > 0$, the occurrence frequency in the payload of a 2_ν-gram $\{\beta_1, \beta_{\nu+2}\}$ can be viewed as a marginal probability computed on the distribution of the $(\nu+2)$-grams that start with β_1 and end with $\beta_{\nu+2}$. In practice the frequency of a 2_ν-gram somehow "summarizes" the occurrence frequency of 256^ν n-grams, with $n = \nu + 2$.

From the occurrence frequency of the n-grams it is possible to derive the distribution of the $(n-1)$-grams, $(n-2)$-grams, etc. On the other hand, measuring the occurrence frequency of the 2_ν-grams does not allow us to automatically derive the distribution of $2_{(\nu-1)}$-grams, $2_{(\nu-2)}$-grams, etc. The distributions of 2_ν-grams with different values of ν give us different structural information about the payload. The intuition is that, ideally, if we could somehow combine the structural information extracted using different values of $\nu = 0, .., N$ we would be able to reconstruct the structural information given by the distribution of n-grams, with $n = (N + 2)$. This motivates the combination of classifiers that work on different descriptions of the payload obtained using the 2_ν-gram technique with different values of ν.

3.2 Feature Reduction

Payload anomaly detection based on the frequency of n-grams is analogous to a text classification problem for which the bag-of-words model and a simple unweighted raw frequency vector representation [17] is used. The different possible n-grams can be viewed as the words, whereas a payload can be viewed as a document to be classified. In general for text classification only the words that are present in the documents of the training set are considered. This approach is not suitable in case of a one-class classification problem. Given that the training set contains (almost) only target examples (i.e., "normal" documents), we cannot conclude that a word that have a probability equal to zero to appear in the training dataset will not be discriminant. As a matter of fact, if we knew of a word w that has probability $p(w|d_t) = 0$, $\forall d_t \in C_t$, of appearing in the class of target documents C_t, and $p(w|d_o) = 1$, $\forall d_o \in C_o$, of appearing in documents of the outlier class C_o, it would be sufficient to measure just one binary feature, namely the presence or not of w_t in the document, to construct a perfect classifier. This is the reason why we choose to take into account all the 256^n n-grams, even though their occurrence frequency measured on the training set is equal to zero. Using the 2_ν-gram technique we still extract 256^2 features. This high number of features could make it difficult to construct an accurate classifier, because of the curse of dimensionality [9] and possible computational complexity problems related to learning algorithms.

In order to reduce the dimensionality of the feature space for payload anomaly detection, we apply a feature clustering algorithm originally proposed by Dhillon et al. in [7] for text classification. Given the number of desired clusters, the algorithm iteratively aggregates the features until the information loss due to the clustering process is less than a certain threshold. This clustering algorithm has the property to reduce the within cluster and among clusters Jensen-Shannon divergence [7] computed on the distribution of words, and has been shown to help obtain better classification accuracy results with respect to other feature reduction techniques for text classification [7]. The inputs to the algorithm are:

1. The set of distributions $\{p(C_i|w_j) : 1 \le i \le m,\ 1 \le j \le l\}$, where C_i is the i-th class of documents, m is the total number of classes, w_j is a word and l is the total number of possible different words in the documents.
2. The set of all the priors $\{p(w_j),\ 1 \le j \le l\}$.
3. The number of desired clusters k.

The output is represented by the set of word clusters $W = \{W_1, W_2, .., W_k\}$. Therefore, after clustering the dimensionality of the feature space is reduced from l to k. In the original l-dimensional feature space, the j-th feature of a pattern vector $\mathbf{x}_i$ represents the occurrence frequency $f(w_j|d_i)$ of the word w_j in the document d_i. The new representation $\mathbf{x}'_i$ of d_i in the k-dimensional feature space can be obtained by computing the features according to

$$f(W_h|d_i) = \sum_{w_j \in W_h} f(w_j|d_i), \quad h = 1, .., k \tag{8}$$

where $f(W_h|d_i)$ can be interpreted as the occurrence frequency of the cluster of words W_h in the document d_i.

In case of a one-class problem, $m = 2$ and we can call C_t the target class and C_o the outlier class. The posterior probabilities $\{p(C_i|w_j) : i = t, o,\ 1 \le j \le l\}$ can be computed as

$$p(C_i|w_j) = \frac{p(w_j|C_i)p(C_i)}{p(w_j|C_t)p(C_t)+p(w_j|C_o)p(C_o)} \qquad i = t, o, \quad 1 \le j \le l \tag{9}$$

and the priors $\{p(w_j),\ 1 \le j \le l\}$ can be computed as

$$p(w_j) = p(w_j|C_t)p(C_t) + p(w_j|C_o)p(C_o),\ \ 1 \le j \le l \tag{10}$$

The probabilities $p(w_j|C_t)$ of finding a word w_j in documents of the target class C_t can be reliably estimated on the training dataset, whereas it is difficult to estimate $p(w_j|C_o)$, given the low number (or the absence) of examples of documents in the outlier class C_o. Similarly, it is difficult to reliably estimate the prior probabilities $p(C_i) = \frac{N_i}{N},\ i = t, o$, where N_i is the number of training patterns of the class C_i and $N = N_t + N_o$ is the total number of training patterns. Given that $N_o \ll N_t$ (or even $N_o = 0$), the estimated priors are $p(C_o) \simeq 0$ and $p(C_t) \simeq 1$, which may be very different from the real prior probabilities.

In our application, the words w_j are represented by the 256^2 possible different 2_ν-grams (with a fixed ν). In order to apply the feature clustering algorithm, we estimate $p(w_j|C_t)$ by measuring the occurrence frequency of the 2_ν-grams w_j on the training dataset and we assume a uniform distribution $p(w_j|C_o) = \frac{1}{l}$ of the 2_ν-grams for the outlier class. We also assume $p(C_o)$ to be equal to the desired rejection rate for the one-class classifiers (see Section 2), and accordingly $p(C_t) = 1 - p(C_o)$.

3.3 Combining One-Class Classifiers

Multiple Classifier Systems (MCS) have been proven to improve classification performaces in many applications [8]. MCS achieve better performance than the best single classifier when the classifiers of the ensemble are accurate and diverse, i.e., make different errors on new patterns [8]. Diversity can be intuitively induced for example by combining classifiers that are based on descriptions of the patterns in different feature spaces [16]. In this paper we use a simple majority voting rule [16] to combine one-class classifiers that work on different descriptions of the payload. Suppose we have a dataset of payloads $T = \{\pi_1, \pi_2, .., \pi_m\}$. Given a payload π_k, we extract the features as discussed in Section 3 obtaining L different descriptions $\{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, .., \mathbf{x}_k^{(L)}\}$ of π_k. L one-class classifier are constructed. The h-th classifier is trained on a dataset $D^{(h)} = \{\mathbf{x}_1^{(h)}, \mathbf{x}_2^{(h)}, .., \mathbf{x}_m^{(h)}\}$, obtained from T using the h-th description for the payloads. During the operational phase, a payload is classified as target (i.e., normal) if it is labeled as target by the majority of the classifiers, otherwise it is classified as outlier (i.e., anomalous).

4 Experiments

In this section we compare and discuss the classification performance of four different anomaly IDS. We compare the performace obtained using the original implementation of 1-gram PAYL [28] developed at Columbia University, an implementation of 2-gram PAYL, and two anomaly IDS we built by combining multiple one-class classifiers. One of these two IDS was implemented using an ensemble of one-class SVM classifiers, whereas the other was implemented using an ensemble of Mahalanobis Distance-based (MD) one-class classifiers. We also show and discuss the performance of the single classifiers used to construct the ensembles. To the best of our knowledge, no public implementation of 2-gram PAYL exists. We implemented our own (simplified) version of 2-gram PAYL in order to compare its performance to the other considered anomaly IDS.

4.1 Experimental Setup

It is easy to see that the accuracy of the anomaly detection systems we consider can be considerably influenced by the values assigned to a number of free parameters. Tuning all the free parameters in order to find the optimal configuration is a difficult and computationally expensive search task. We did not perform a complete tuning of the parameters, but we used a number of reasonable values that should represent an acceptable suboptimal configuration. For 1-gram PAYL we used the default configuration provided along with the software. For all the MD classifiers and our 2-gram PAYL we set the smoothing factor $\alpha = 0.001$, because this is the same default value for α used by 1-gram PAYL (which also uses the MD classification algorithm). We used `LibSVM` [4] to perform the experiments with one-class SVM. For all the one-class SVM classifiers we used the gaussian kernel in Equation (1). In order to choose a suitable value for γ we performed a number of pilot experiments. We noted that setting $\gamma = 0.5$ the one-class SVM classifiers performed well in all the different feature spaces obtained by varying the parameters ν and the number of feature clusters k during the feature extraction and reduction processes, respectively (see Section 3.1). Having fixed the values for some of the parameters as explained above, we performed several experiments varying the "gap" ν and the number of feature clusters k. The values we used for this parameters and the obtained results are discussed in detail in Section 4.2.

We performed all the experiments using 5 days of HTTP requests towards our department's web server collected during October 2004. We assumed this unlabeled traffic to contain mainly normal requests and possibly a low fraction of noise, i.e., anomalous packets. We used the first day of this traffic to train the IDS and the last 4 days to measure the false positive rate (i.e., the false alarm rate). In the following we refer to the first day of traffic as *training dataset*, and to the last 4 days as *test dataset*. The training dataset contained 384,389 packets, whereas the test dataset contained 1,315,433 packets. In order to estimate the detection rate we used 18 HTTP-based buffer overflow attacks. We collected the first 10 attacks from the Internet (e.g., exploits for `IIS 5.0 .printer ISAPI Extension` [1], `ActivePerl perlIIS.dll` [2], among others). Each of these attacks is made up of a different number of attack packets. The latter 8 attacks were represented by some of the attacks used in [12], where Fogla et al. constructed a number of mimicry attacks against PAYL. These attacks were derived from an exploit that targets a vulnerability in `Windows Media Services (MS03-022)` [11]. In particular, we used the original `Windows Media Services` exploit used in [12] before transformation, 6 mimicry attacks derived from this original attack using a polymorphic shellcode engine called CLET [6], and one polymorphic blending attack obtained using the *single byte encoding* scheme for the 2-grams presented in [12]. The 6 mimicry attacks obtained using CLET were created setting different combinations of packet length and total number of attack packets. The polymorphic blending attack consisted of 3 attack packets and the payload of each packet was 1460 bytes long. In the following we will refer to the set of attacks described above as *attack dataset*. Overall, the attack dataset contained 126 attack packets.

4.2 Performance Evaluation

In order to compare the performace of PAYL, the constructed single classifiers, and the overall anomaly IDS, we use the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC). We trained PAYL and the single classifiers for different operational points, i.e., we constructed different "versions" of the classifiers setting a different rejection rate on the training dataset each time. This allowed us to plot an approximate ROC curve for each classifier. Assuming the training dataset contains only normal HTTP requests, the rejection rate can be interpreted as a *desired false positive* rate. In the following we refere to this desired false positive rate as DFP. If we also assume the test dataset contains only normal HTTP requests, we can use it to estimate the *"real" false positive* rate, or RFP. Each point on an ROC curve represents the RFP and the detection rate (DR) produced by the classifier. The detection rate is measured on the attack dataset and is defined as the faction of detected attack packets, i.e., the number of attack packets that are classified as anomalous divided by the total number of packets in the attack dataset (regardless of the specific attack the detected packets come from).

We measured the performance of the classifiers for 7 different operational points to compute an (partial) ROC curve for each classifier. These points are obtained by training each classifier using 7 DFP, namely 0%, 0.01%, 0.1%, 1.0%, 2.0%, 5.0% and 10.0%. The AUC is estimated by integrating the ROC curve in the interval of RFP between 0% and 10.0%. The obtained result is then normalized so that the maximum possible value for the AUC is 1. According to how the AUC is computed, the higher the value of the

DFP(%)	RFP(%)	Detected attacks	DR(%)
0.0	0.00022	1	0.8
0.01	0.01451	4	17.5
0.1	0.15275	17	69.1
1.0	0.92694	17	72.2
2.0	1.86263	17	72.2
5.0	5.69681	18	73.8
10.0	11.05049	18	78.6

Table 1: Performance of 1-gram PAYL.

		k				
		10	**20**	**40**	**80**	**160**
ν	**0**	0.9660 (0.4180E-3)	0.9664 (0.3855E-3)	0.9665 (0.4335E-3)	0.9662 (0.2100E-3)	**0.9668** (0.4686E-3)
	1	0.9842 (0.6431E-3)	0.9839 (0.7047E-3)	**0.9845** (0.7049E-3)	0.9833 (1.2533E-3)	0.9837 (0.9437E-3)
	2	0.9866 (0.7615E-3)	0.9867 (0.6465E-3)	0.9875 (0.6665E-3)	**0.9887** (2.6859E-3)	0.9862 (0.7753E-3)
	3	0.9844 (1.2207E-3)	0.9836 (1.1577E-3)	**0.9874** (1.0251E-3)	0.9832 (1.0619E-3)	0.9825 (0.6835E-3)
	4	0.9846 (0.5612E-3)	0.9847 (1.5334E-3)	0.9846 (0.9229E-3)	0.9849 (1.5966E-3)	**0.9855** (0.4649E-3)
	5	0.9806 (0.8638E-3)	0.9813 (0.9072E-3)	0.9810 (0.5590E-3)	0.9813 (0.8494E-3)	**0.9818** (0.3778E-3)
	6	0.9809 (0.7836E-3)	0.9806 (1.1608E-3)	**0.9812** (1.6199E-3)	0.9794 (0.3323E-3)	0.9796 (0.4240E-3)
	7	0.9819 (1.6897E-3)	0.9854 (0.8485E-3)	0.9844 (1.2407E-3)	0.9863 (1.9233E-3)	**0.9877** (0.7670E-3)
	8	0.9779 (1.7626E-3)	0.9782 (1.9797E-3)	0.9787 (2.0032E-3)	**0.9793** (1.0847E-3)	0.9785 (1.7024E-3)
	9	0.9733 (3.1948E-3)	**0.9775** (1.9651E-3)	0.9770 (1.0803E-3)	0.9743 (2.4879E-3)	0.9722 (1.2258E-3)
	10	0.9549 (2.7850E-3)	0.9587 (3.3831E-3)	0.9597 (3.8900E-3)	0.9608 (1.2084E-3)	**0.9681** (7.1185E-3)

Table 2: Performance of single one-class SVM classifiers. The numbers in bold represent the best average AUC for a fixed value of ν. The standard deviation is reported between parentheses.

AUC, the better the performance of the classifier in the considered interval of false positives. For each DFP, we also measured the number of detected attacks. We consider an attack as detected if at least one out of the total number of packets of the attack is detected as anomalous. It is worth noting that the number of detected attacks is different from the detection rate used to computed the ROC curve.

1-gram PAYL. Our baseline is represented by the performance of 1-gram PAYL. As mentioned before, PAYL measures the occurrence frequency of byte values in the payload. A separate model is generated for each different payload length. These models are clustered together at the end of the training to reduce the total number of models. Furthermore, the length of a payload is also monitored for anomalies. Thus, a payload with an unseen or very low frequency length is flagged as an anomaly [28].

We trained PAYL using the entire first day of collected HTTP requests. We constructed the ROC curve by estimating the RFP on the entire test dataset, i.e., the other 4 days of collected HTTP requests, and the detection rate on the attack dataset. The obtained AUC was equal to 0.73. As shown in Table 1, for DFP=0.1% PAYL produced an RFP=0.15% and was able to detect 17 out of 18 attacks. In particular it was able to detect all the attacks except the polymorphic blending attack. Table 1 also shows that the polymorphic blending attack remained undetected until RFP is around 5.7%. By performing further experiments, we found out that the minimum amount of RFP for which PAYL is able to detect all the attacks, included the polymorphic blending attack, is equal to 4.02%, which is usually considered intolerably high for network intrusion detection.

Single One-Class SVM Classifiers. We constructed several one-class SVM classifiers. We extracted the features as described in Section 3.1 varying the parameter ν from 0 to 10, thus obtaining 11 different descriptions of the patterns. Then, for each fixed ν, we applied the feature clustering algorithm described in Section 3.2 fixing the prior probability $P(C_o) = 0.01$ and setting the number of desired clusters k equal to 10, 20, 40, 80 and 160. We used a random initialization for the algorithm (i.e., at the first step each feature is randomly assigned to one of the k clusters). The feature clustering algorithm stops when the information loss due to the feature clustering becomes minor than 10^{-4}.

For each pair (ν, k) of parameter values we repeated the experiment 5 times. For each round we applied the feature clustering algorithm (using a new random initialization), and we trained a classifier on a sample of the training dataset obtained from the original training dataset by applying the bootstrap technique without replacement and with a sampling ratio equal to 10%. We estimated the AUC by measuring the false positives on a sample of the test dataset obtained using again the bootstrap technique with sampling ratio equal to 10%, and measuring the detection rate on the entire attack dataset. Table 2 reports the estimated average AUC. The numbers between parentheses represent the standard deviation computed over the 5 rounds. We discuss the obtained results later in this section comparing them to the results obtained using the MD classification algorithm.

Single MD Classifiers. Similarly to the experiments with the one-class SVM classifiers, we constructed several MD classifiers. For each pair (ν, k) of parameter values, we applied the feature clustering algorithm with random initialization, and we trained a classifier on a 10% sample of the training set (using again the bootstrap technique without replacement). The AUC was estimated by measuring the false positives on a 10% sample of the test dataset and the detection rate on the entire attack dataset. We repeated each experiment 5 times. Table 3 reports the average and the standard deviation for the obtained AUC. The MD classifier performs extremely well for $\nu = 0$ and $k = 10$. In this case the MD classifier is able to detect all of the 18 attacks for an RFP around 0.1% and reaches 100% of detection rate for an RFP around 1%. However, the use of only one classifier

				k		
		10	**20**	**40**	**80**	**160**
ν	**0**	**0.9965** (0.5345E-3)	0.9948 (1.4455E-3)	0.9895 (3.9813E-3)	0.9785 (5.1802E-3)	0.9718 (9.9020E-3)
	1	**0.9752** (0.5301E-3)	0.9729 (0.7921E-3)	0.9706 (1.0940E-3)	0.9664 (2.2059E-3)	0.9653 (0.3681E-3)
	2	**0.9755** (0.2276E-3)	0.9743 (0.4591E-3)	0.9741 (0.9121E-3)	0.9676 (0.1084E-3)	0.9661 (0.4246E-3)
	3	**0.9749** (0.7496E-3)	0.9736 (0.8507E-3)	0.9726 (1.8217E-3)	0.9714 (1.2729E-3)	0.9708 (2.6994E-3)
	4	**0.9761** (0.4269E-3)	0.9743 (0.3552E-3)	0.9735 (0.7998E-3)	0.9737 (0.3827E-3)	0.9722 (0.9637E-3)
	5	**0.9735** (1.0645E-3)	0.9692 (0.3607E-3)	0.9694 (1.0499E-3)	0.9626 (2.4574E-3)	0.9606 (1.9866E-3)
	6	**0.9737** (0.6733E-3)	0.9709 (1.5523E-3)	0.9687 (2.9730E-3)	0.9699 (4.1122E-3)	0.9717 (0.5427E-3)
	7	**0.9687** (3.3302E-3)	0.9545 (9.6519E-3)	0.9505 (7.3100E-3)	0.9258 (19.923E-3)	0.8672 (50.622E-3)
	8	**0.9731** (0.7552E-3)	0.9721 (0.6001E-3)	0.9717 (0.6799E-3)	0.9715 (0.6367E-3)	0.9678 (1.5209E-3)
	9	**0.9719** (1.5743E-3)	0.9695 (1.9905E-3)	0.9700 (2.2792E-3)	0.9662 (2.9066E-3)	0.9611 (1.5542E-3)
	10	0.9641 (1.6604E-3)	**0.9683** (2.5370E-3)	0.9676 (1.2692E-3)	0.9635 (1.1016E-3)	0.9598 (0.6209E-3)

Table 3: Performance of single MD classifiers. The numbers in bold represent the best average AUC for a fixed value of ν. The standard deviation is reported between parentheses.

					ν					
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.9744	0.9665	0.9711	0.9393	0.9170	0.8745	0.8454	0.8419	0.8381	0.9556	0.9079

Table 4: Performance of single MD classifiers for varying ν. No feature clustering is applied. The number in bold represents the best result.

does not improve the hardness of evasion, as discussed in Section 4.3.

We also estimated the performance of the MD classifiers without applying the feature clustering algorithm. In this case each pattern is described by 65,536 features. We trained a classifier for each value of $\nu = 0,..,10$ on the entire training dataset and estimated the AUC measuring the false positives and the detection rate on the entire test and attack dataset, respectively. The obtained results are reported in Table 4. As can be seen from Table 3 and Table 4, the best performance for a fixed value of ν is always reached using $k = 10$. The only exception is when $\nu = 10$. In this case the best performance is obtained using $k = 20$. The good performance obtained for low values of k is probably due to the fact that the MD classification algorithm suffers from the curse of dimensionality problem. By reducing the dimensionality of the feature space the MD classifier is able to construct a tighter decision surface around the target class. For each fixed k the best results in terms of AUC were obtained using $\nu = 0$. The only exception is when $k = 160$. In this case the best AUC is obtained for $\nu = 4$. Nevertheless, the AUC obtained for $\nu = 4$ and for $\nu = 0$ are really close, and considering the standard deviation it is not possible to say which classifier performs better than the other. As we discuss in Section 4.3, the amount of structural information extracted from the payload decreases when ν grows. The MD classifier seems to be sensitive to this effect.

By comparing the best results in Table 2 and Table 3 (the numbers in bold), it is easy to see that SVM classifiers perform better than MD classifiers in all the cases except when $\nu = 0$ and $\nu = 10$. When $\nu = 10$ the best performance are really close, and considering the standard deviation it is not possible to say which classifier performs better than the other. It is also easy to see that, differently from the MD classification algorithm, the one-class SVM seems not to suffer from the growing dimensionality of the feature space obtained by increasing k. This is probably due to the fact that by using the gaussian kernel the patterns are projected in an infinite-dimensional feature space, so that the dimensionality of the original feature space becomes less important.

2-gram PAYL. The MD classifier constructed without applying the feature clustering and setting $\nu = 0$ represents an implementation of 2-gram PAYL that uses one model for all the possible packet lengths. Table 5 reports the results obtained with this classifier. It is easy to see that 2-gram PAYL performs better that 1-gram PAYL, if we consider the detection rate DR. This is due to the fact that the simple distribution of 1-grams (i.e., the distribution of the occurrence frequency of the byte values) does not extract structural information from the payload, whereas the distribution of 2-grams conveys byte sequence information. Nevertheless, 2-gram PAYL is not able to detect the polymorphic blending attack even if we are willing to tolerate an RFP as high as 11.25%. This is not surprising given that the polymorphic blending attack we used was specifically tailored to evade 2-gram PAYL.

Classifier Ensembles. We constructed several anomaly IDS by combining multiple classifiers using the simple majority voting rule. We first combined one-class SVM classifiers. For a fixed value of the number of feature clusters k, the output of the 11 classifiers constructed for $\nu = 0,..,10$ were combined. The experiments were repeated 5 times for each value of k. We also applied the same approach to

DFP(%)	RFP(%)	Detected attacks	DR(%)
0.0	0.00030	14	35.2
0.01	0.01794	17	96.0
0.1	0.12749	17	96.0
1.0	1.22697	17	97.6
2.0	2.89867	17	97.6
5.0	6.46069	17	97.6
10.0	11.25515	17	97.6

Table 5: Performance of an implementation of 2-gram PAYL using a single MD classifier, $\nu = 0$ and $k = 65,536$.

combine MD classifiers. The average and standard deviation for the obtained AUC are reported in Table 6. The last row reports the results obtained by combining single MD classifiers for which no feature clustering was applied (i.e., all the 65,536 features are used). The combination works really well in case of one-class SVM. As shown in Table 6, the overall IDS constructed using ensembles of one-class SVM always performs better than the best single classifier. The only exception is when $k = 160$, but in this case the results are so close that considering the standard deviation it is not possible to say which one is the best. On the other hand, the combination of MD classifiers is not as effective as for the ensemble of one-class SVM, and does not improve the performance of the single best classifier. This is probably due to the fact that although we constructed MD classifiers that work on different feature spaces, the obtained classifiers are not sufficiently diverse and make the same errors for new patterns.

k	Ensemble of SVM	Ensemble of MD
10	0.9885 (0.3883E-3)	**0.9758** (0.4283E-3)
20	0.9875 (2.0206E-3)	0.9737 (0.1381E-3)
40	**0.9892** (0.2257E-3)	0.9736 (0.2950E-3)
80	0.9891 (1.6722E-3)	0.9733 (0.5144E-3)
160	0.9873 (0.4209E-3)	0.9701 (0.6994E-3)
65,535	-	0.9245

Table 6: Average AUC of classifier ensembles constructed using the majority voting rule. The numbers in bold represent the best result for varying k. The standard deviation is reported between parentheses.

DFP(%)	RFP(%)	Detected attacks	DR(%)
0.0	0.0	0	0
0.01	0.00381	17	68.5
0.1	0.07460	17	79.0
1.0	0.49102	18	99.2
2.0	1.14952	18	99.2
5.0	3.47902	18	99.2
10.0	7.50843	18	100

Table 7: Performance of an overall IDS constructed using an ensemble of one-class SVM and setting $k = 40$. The DFP is referred to the single classifiers of the ensemble.

Table 7 shows the results obtained with an overall IDS implemented by combining the 11 single one-class SVM constructed using $\nu = 0, .., 10$ and $k = 40$. The IDS is able to detect all the attacks except the polymorphic blending attack for an RFP lower than 0.004%. The IDS is also able to detect all the attacks, including the polymorphic blending attack, for an RFP lower than 0.5%.

In conclusion, the experimental results reported above show that our IDS constructed using an ensemble of one-class SVM classifiers and using $k = 40$ performs better than any other IDS or single classifiers we considered. The only exception is the single MD classifier obtained setting $\nu = 0$ and $k = 10$. However, as mentioned before and as discussed in Section 4.3, this single MD classifier may still be easy to evade, whereas our MCS based IDS is much harder to evade.

4.3 Discussion and Future Work

2_ν-grams. We discussed in Section 3.1 how to extract the features using the 2_ν-gram technique. We also argued that the occurrence frequency of 2_ν-grams somehow "summarizes" the occurrence frequency of n-grams. This allows us to capture some byte sequence information. In order to show that the 2_ν-grams actually extract structural information from the payload, we can consider the bytes in the payload as random variables and then we can compute the relative mutual information of bytes that are ν positions apart from each other. That is, for a fixed value of ν we compute the quantity

$$RMI_{\nu,i} = \frac{I(B_i; B_{i+\nu+1})}{H(B_i)} \tag{11}$$

where $I(B_i; B_{i+\nu+1})$ is the mutual information of the bytes at position i and $(i+\nu+1)$, and $H(B_i)$ is the entropy of the bytes at position i. By computing the average for $RMI_{\nu,i}$ over the index $i = 1, .., (L-\nu-1)$, with L equal to the maximum payload length, we obtain the average relative mutual information for the 2_ν-grams along the payload. We measured this average relative mutual information on both the training and the test set varying ν from 0 to 20. The results are shown in Figure 1. It is easy to see that the amount of information extracted using the 2_ν-gram technique is maximum for $\nu = 0$ (i.e., when the 2-gram technique is used) and decreases for growing ν. However the decreasing trend is slow and the average RMI is always higher than 0.5 until $\nu = 10$. This is probably due to the fact that HTTP is a highly structured protocol. Preliminary results show that the same property holds for other text based protocols.

Polymorphic Blending Attack. The polymorphic blending attack we used for our performance evaluation was presented in [12] as an attack against 2-gram PAYL. The polymorphic blending attack encodes the attack payload so that the distribution of 2-grams in the transformed attack

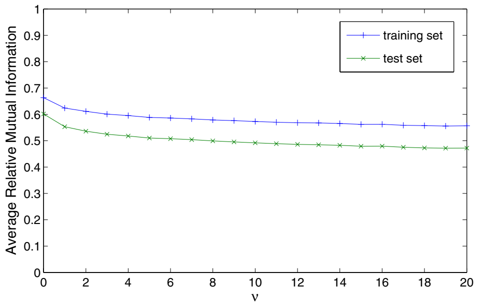

Figure 1: Average relative mutual information for varying ν.

"looks" like normal, from the point of view of the model of normal traffic constructed by PAYL. As discussed in [12], a polymorphic blending attack against 2-gram PAYL is also able to evade 1-gram PAYL. This is because the distribution of 1-grams can be derived from the distribution of 2-grams. Thus, if the distribution of 2-grams in the attack payload "looks" like normal, so does the distribution of 1-grams.

In order to construct the attack, first of all the attacker needs to monitor part of the traffic towards the network protected by the IDS [12]. By monitoring this traffic, a polymorphic blending engine constructs an approximate normal profile for the 2-grams and transforms the attack payload accordingly. It has been proven that a "perfect" single byte encoding transformation of the attack payload in order to reflect the estimated normal profile is NP-complete [12]. Therefore, Fogla et al. [12] proposed an approximate solution to the problem. High frequency 2-grams in the attack payload are greedily matched via one-to-one byte substitution with 2-grams that have high frequencies in normal traffic. The proposed approach could also be generalized to evade an n-gram version of PAYL. However, because of the way the algorithm greedily matches n-grams in the attack payload with n-grams in normal traffic [12], the single byte encoding algorithm proposed is less and less likely to generate a successful attack payload transformation, as n grows. This means that although the polymorphic blending attack may still work well for $n = 2$, it is likely to fail for $n \gg 2$.

Hardness of Evasion. In Section 4.2 we showed that an MD classifier constructed using $\nu = 0$ (i.e., using the 2-gram technique) and $k = 10$ achieves very good classification performance (see Table 3). However, the use of only one classifier does not help in hardening the anomaly detector against evasion attempts. The attacker may easily modify the polymorphic blending attack against 2-gram PAYL in order to evade this one particular classifier.

We constructed our anomaly IDS using multiple classifiers that work on different descriptions of the payload. In this case the polymorphic blending attack that mimics the normal distribution of 2-grams does not work anymore because it can already be detected for a percentage of false positives as low as 0.5%, as shown by the experimental results reported in Section 4.2. In order for the attacker to evade our IDS, she needs to devise a substitution algorithm that evades the majority of the classifiers at the same time. Therefore, the attacker needs to transform the attack payload in order to mimic the distribution of 2_ν-grams for different values of ν. Because of the way the features are extracted using the 2_ν-gram technique, this result may be achieved by a polymorphic transformation that encodes the attack payload to reflect the distribution of the n-grams in normal traffic, with n greater than $\frac{max(\nu)+2}{2}$. Here $max(\nu)$ represents the maximum value of ν used during the feature extraction process. Thus, in order to evade our IDS the attacker needs to encode the attack payload mimicking the distribution in normal traffic of 7-grams. This makes it much harder to evade our IDS, compared to 1-gram and 2-gram PAYL. In theory, a hypothetical 7-gram PAYL would be as hard to evade as our IDS. However, this hypothetical 7-gram PAYL would easily suffer from the curse of dimensionality and memory consumption problems due to the huge number of features (equal to 256^7). Our anomaly IDS is much more practical.

Future Work. We leave a thorough run-time performance comparison between PAYL and our ensemble of one-class SVM classifiers to future work. However, it is worth noting that during the operational phase both the feature extraction, feature clustering, and the test of a new pattern against the SVM models can be efficiently performed. For these reasons, we expect the overhead introduced by an optimized version of our IDS to be sufficiently low. We also plan to study the application of different classifier combination rules (e.g., average and product [16, 24]) and to compare them against the results obtained using the simple majority voting rule.

5 Conclusion

In this paper we constructed a payload-based anomaly IDS by using an ensemble of one-class SVM classifiers. We proposed a new technique to extract the features from the payload inspired by the n-gram analysis technique. The proposed technique allowed us to obtain descriptions of the payload in different feature spaces. We constructed several one-class SVM classifiers. Each classifier works on a different description of the payload. The experimental results show that the combination of the obtained classifiers improves both the detection accuracy and the hardness of evasion with respect to other recently proposed payload-based anomaly IDS.

Acknowledgments

This work is supported in part by NSF grant CCR-0133629 and Office of Naval Research grant N000140410735. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of NSF and the U.S. Navy.

The authors would like to thank Prahlad Fogla for the help in constructing the polymorphic attacks, and Giorgio Giacinto, Fabio Roli and the anonymous reviewers for their insightful comments.

References

[1] Securityfocus (BID 2674). Microsoft IIS 5.0 .printer ISAPI Extension buffer overflow vulnerability, 2006.

[2] Securityfocus (BID 3526). ActivePerl perlIIS.dll buffer overflow vulnerability, 2006.

[3] R. Brunelli and D. Falavigna. Person identification using multiple cues. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(10):955–966, 1995.

[4] C. Chang and C. Lin. LIBSVM: a library for support vector machines, 2001.

[5] R. Chinchani and E.V.D. Berg. A fast static analysis approach to detect exploit code inside network flows. In *Recent Advances in Intrusion Detection (RAID)*, 2005.

[6] T. Detristan, T. Ulenspiegel, Y. Malcom, and M. Underduk. Polymorphic shellcode engine using spectrum analysis. *Phrack Issue 0x3d*, 2003.

[7] I. S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.

[8] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems (MCS)*, 2000.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2000.

[10] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In D. Barbara and S. Jajodia, editors, *Applications of Data Mining in Computer Security*. Kluwer, 2002.

[11] Firew0rker. Windows Media Services remote command execution exploit, 2006.

[12] P. Fogla, M. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee. Polymorphic blending attack. In *USENIX Security Symposium*, 2006.

[13] G. Giacinto, F. Roli, and L. Didaci. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recognition Letters*, 24(12):1795–1803, 2003.

[14] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Automating mimicry attacks using static binary analysis. In *USENIX Security Symposium*, 2005.

[15] C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *ACM Symposium on Applied Computing (SAC)*, 2002.

[16] L. Kuncheva. *Combining Pattern Classifiers*. Wiley, 2004.

[17] E. Leopold and J. Kindermann. Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46:423–444, 2002.

[18] M. Mahoney. Network traffic anomaly detection based on packet bytes. In *ACM Symposium on Applied Computing (SAC)*, 2003.

[19] J. McHugh, A. Christie, and J. Allen. Defending yourself: The role of intrusion detection systems. *IEEE Software*, pages 42–51, Sept./Oct. 2000.

[20] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[21] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and RC Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[22] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[23] D. M. J. Tax. *One-Class Classification, Concept Learning in the Absence of Counter Examples*. PhD thesis, Delft University of Technology, Delft, Netherland, 2001.

[24] D. M. J. Tax and R. P. W. Duin. Combining one-class classifiers. In *Multiple Classifier Systems (MCS)*, 2001.

[25] T. Toth and C. Kruegel. Accurate buffer overflow detection via abstract payload execution. In *Recent Advances in Intrusion Detection (RAID)*, 2002.

[26] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[27] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *ACM Conference on Computer and Communication Security (ACM CCS)*, 2002.

[28] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection (RAID)*, 2004.

[29] K. Wang and S. Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advances in Intrusion Detection (RAID)*, 2005.