Unsupervised Learning Algorithms for Intrusion Detection

Stefano Zanero
Dip. di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano Italy
Email: zanero@elet.polimi.it

Giuseppe Serazzi
Dip. di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano Italy
Email: serazzi@elet.polimi.it

***Abstract*—This work summarizes our research on the topic of the application of unsupervised learning algorithms to the problem of intrusion detection, and in particular our main research results in network intrusion detection. We proposed a novel, two tier architecture for network intrusion detection, capable of clustering packet payloads and correlating anomalies in the packet stream. We show the experiments we conducted on such architecture, we give performance results, and we compare our achievements with other comparable existing systems.**

I. Introduction

One of the most daunting tasks in the operations and management of large and complex networks today is ensuring their overall security. One of the key processes in the management of network security is the detection of security incidents, followed by identification and appropriate reaction. *Intrusion Detection Systems* [1], [2] are designed to help network administrators in this process. While such systems have been the subject of much research, the majority of network-based IDS systems deployed today are misuse-based, meaning that they use a knowledge base to recognize directly the *signatures* of intrusion attempts. Therefore, misuse detectors are powerless against new, "zero-day" attacks, and they are subject to a wide array of evasion techniques [3].

Anomaly detection systems, on the other hand, try to create a model of normal behavior, and flag any deviation as suspicious. Not requiring "a priori" knowledge of the attacks, they are theoretically able to detect any type of misbehavior in a statistical way; however, they tend to be more difficult to design and to deploy than misuse detectors, and they are rarely used in practical applications. Our research work focused on the use of *unsupervised learning algorithms* for approaching the problem of anomaly detection in complex systems, since we believe that the developments in learning algorithms, the availability of computational power, and the new trends in information security make new research in the field of anomaly detection extremely promising.

In the thesis which we are summarizing in this paper [4], we describe our implementation of both a network-based and a host-based anomaly detection IDS. We will focus here on the network-based prototype. Our key original contributions in the thesis can be identified as follows:

- We proposed a two-tier architecture to analyze network packets overcoming the dimensionality problems which arise in the application of unsupervised learning techniques to network-based anomaly detection.
- We proposed improvements and heuristics to increase the throughput of Self Organizing Maps to a rate suitable for online Intrusion Detection purposes.
- We carefully evaluated the detection rate and the false positive rate against well known datasets, comparing the results with state of the art systems described in previous literature.

The remainder of this work is organized as follows. In Section II we describe our two-tier approach, and compare it with earlier works. In Section III we describe the algorithms we used for clustering TCP/IP payloads. In Section IV we describe the algorithm used for outlier detection in multivariate time series. In Section V we describe our experimental evaluation. In Section VI we draw our conclusions and outline recent developments of our work.

II. A two-tier architecture for Network Intrusion Detection

The problem of network intrusion detection can be reformulated, in the unsupervised learning framework, as the detection of anomalies in the flow of packets (or connections), on a TCP/IP network. While the packet header data can be mapped onto a multivariate time series, the varying size of the payload data and its heterogeneous nature defy a compact representation as a single feature. Most existing research on the use of unsupervised learning algorithms for network intrusion detection avoid this problem altogether by discarding the payload and retaining only the information in the packet header [5]–[9].

Ignoring the payload of packets, however, inevitably leads to information loss: most attacks, in fact, are detectable only by analyzing the payload of a packet, not the headers alone. Despite their reduced coverage, these algorithms show interesting, albeit obviously limited, intrusion detection properties.

Some earlier works tried to deal with this problem: [10] uses a rule-based algorithm to evaluate the payloads but, on the contrary, ignores totally the meaning of the header fields; ALAD [11] detects "keywords" in the protocols in a rather

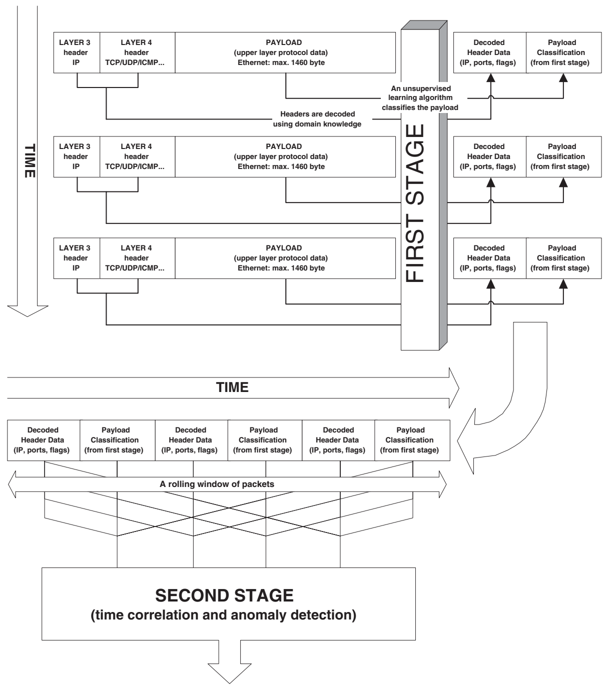
TIME
LAYER 3 header IP
LAYER 4 header TCP/UDP/ICMP...
PAYLOAD (upper layer protocol data) Ethernet: max. 1460 byte
Decoded Header Data (IP, ports, flags)
Payload Classification (from first stage)
An unsupervised learning algorithm classifies the payload
Headers are decoded using domain knowledge
FIRST STAGE
TIME
A rolling window of packets
SECOND STAGE (time correlation and anomaly detection)

Fig. 1. Scheme of the overall architecture of the network-based IDS

limited manner; PAYL [12] uses statistical techniques on the payloads, ignoring the headers.

In order to retain some of the information contained in the payload, while keeping the problem tractable, we proposed a novel architecture for building a network-based anomaly detector using only unsupervised learning algorithms (described originally in [13]). The two-tier architecture we proposed is shown in Figure 1. In the first tier of the system, an unsupervised clustering algorithm operates a basic form of pattern recognition on the payload of the packets, observing one packet payload at a time and "compressing" it into a byte of information (a "payload class" value). This classification can then be added to the information decoded from the packet header (or to a subset of this information), and passed on to the second tier.

The second tier algorithm instead takes into consideration the anomalies, both in each single packet and in a sequence of packets. It is worth noting that most of the solutions proposed by previous researchers in order to analyze the sequence of data extracted by the packet headers could be used as a second tier algorithm, complemented by our first tier of unsupervised pattern recognition and clustering.

III. A Payload Clustering Algorithm

In the first tier we need to find an algorithm that receives as an input the payload of a packet. On an Ethernet segment this means up to 1460 byte values which can be interpreted as *pattern vectors* of variable size. This algorithm must classify these vectors in a *sensible* way. By sensible we mean that the transformation should preserve as much information as possible about the "similarity" between packets; it should separate, as much as possible, packets from different protocols in different groups; most importantly, it should also separate, as much as possible, anomalous or malformed payloads from normal payloads.

This is a typical problem of clustering, i.e., trying to group objects in classes, so that intra-class similarity is maximized and inter-class similarity is minimized [14]. It can also be seen as an instance of a pattern recognition problem applied to packet payloads [15].

There is an endless variety of clustering algorithms, but analyzing the requirements of our problem helps in narrowing the choices. Firstly, many algorithms need a criterion to define a correct or acceptable number of classes, but we do not have such information. We also need an algorithm that, once trained, can be progressively applied to incoming data online. Finally, we need an algorithm which is robust w.r.t. the presence of outliers in training data. An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism [16].

Out of the algorithms that fit these requirements, we tested three widely used approaches: K-means, which is centroid-based; the Principal Direction Divisive Partitioning (PDDP), a hierarchical divisive approach [17]; and Kohonen's Self Organizing Maps (SOM) [18], which are a competitive, hard neural algorithm. Our results [13] show that the SOM algorithm is able to sensibly cluster payload data, discovering interesting information in an unsupervised manner. Additionally, it is robust with regard to the choice of the number of clusters, and it is also resistant to the presence of outliers in the training data.

The computational complexity of unsupervised learning algorithms scales up steeply with the number of considered features, and the detection capabilities decrease correspondingly (this is usually called the "curse of dimensionality"). This effect is particularly strong in our problem, where we analyze vectors containing up to 1460 bytes of data. The throughput of a straightforward C implementation of the Kohonen algorithm on our hardware and software configuration[1] is on average of 3400 packets per second, which is enough to handle a 10 Mb/s Ethernet network, but insufficient for a 100 Mb/s network. A traditional approach to the problem would use dimension reduction techniques such as Principal Component Analysis [19]. But our early experiments demonstrated that such techniques are quite ineffective in this particular situation, since by their nature they tend to "compress" outliers onto normal data, which is exactly the opposite of what we want to achieve.

Thus, we developed several approximate techniques to speed up the SOM algorithm at runtime, by introducing minimal errors in the classification [20]. Let N be the number of classes, and d the number of dimensions of the data. At

[1]The reference machine for our tests is an Athlon-XP 3200 based computer with 1 GB of DDR RAM, running GNU/Linux with a 2.6 kernel. All the tests refer to a SOM with square topology, and a size in the space of neurons of 10×10.

TABLE I
THROUGHPUT AND ERRORS DURING RUNTIME PHASE, WITH VARIOUS OPTIMIZATIONS.

Bytes	Heuristics	K	m	Packets/sec.	Error %
1460	None	-	-	3464.81	-
1460	K-means	10	-	8724.65	0.8
1460	K-means+	5	10	5485.95	0.4
1460	K-means+	10	10	10649.20	0.8
800	None	-	-	4764.11	-
800	K-means+	5	10	9528.26	0.5
800	K-means+	10	10	15407.36	1.0
400	None	-	-	8400.45	-
400	K-means+	5	10	28965.84	0.6
400	K-means+	10	10	30172.65	1.2
200	None	-	-	10494.87	-
200	K-means+	5	10	51724.70	0.8
200	K-means+	10	10	65831.45	2.3

runtime, the SOM requires an order of $N \cdot d$ computations for finding the best match. Instead, we group the N centroids in $K < N$ super-clusters, and then we select the winning neuron in a two-step procedure (the super-cluster first, and then the specific neuron). The algorithm is heuristic, since it can happen that the best matching neuron is not in the best matching super-cluster, but as we will see the error rate is very low. We use K-means among the neurons to form the super-clusters, but we must correct it to overcome initialization dependency and to balance clusters. We re-run the algorithm a fixed number m of times, and choose the distribution in classes which minimizes the average expected number of operations. In the following we denote this variant as "K-means+". This heur istic cannot be easily applied to the training phase, due to the continuous change in the position of the neurons.

In Table I we report the runtime throughput of the algorithm, evaluated in packets per second, depending on different combination of the parameters. They are computed over a window of 10^6 packets. The values are averages over multiple runs of the algorithm on different portions of the dataset. Using a well known study of the statistical properties of Internet traffic [21], it can be seen that the original SOM algorithm, considering the full payload of 1460 maximum bytes per packet, with no heuristics, operates at a speed that is acceptable for use on a 10 Mb/s Ethernet network. Introducing K-means+, we obtain a better throughput with an acceptable error rate. A speed of 10.500 packets/second is enough to handle a normal 100 Mb/s link (considering also the presence of empty packets). It can be also shown that the use of our modified algorithm does not diminish the detection capabilities of the system. We refer the reader to [4] for a detailed demonstration.

Another problem which arises when using similarity-based algorithms in high-dimensional space, is the choice of a good similarity criterion. The two most used distance criteria in SOM literature are the inner product and the euclidean metric. Since the inner product is closely related to the so-called cosine distance, it is particularly useful in those cases where attributes have values whose characteristic is to be either zero or nonzero. We have a range of discrete values instead, so we resorted to the euclidean distance. While this choice has no specific theoretical support, our experiments have shown that it works well. More work could be done to study other, maybe better suited, distance functions.

In recent works the effect of the curse of dimensionality on the concept of "distance metrics" has been studied in detail. In high dimensional spaces such as the one we are considering, the data become very sparse. In [22], [23] it is shown that in high dimensional spaces the concept of proximity and distance may not be meaningful, even qualitatively. However, as most of the hypotheses of such theoretical works do not hold for our variables, we have experimentally observed that in our setup this effect does not happen. We explored the application of the alternative distance metrics proposed in [23], [24], but in our particular application they seem to lump all the data in a few cluster, diminishing the overall recognition capabilities of the algorithm instead of enhancing it. We are currently studying the applicability of wavelet-based distance metrics such as the ones proposed in [25].

IV. Multivariate Time Series Outlier Detection

The second tier algorithm must detect anomalies and outliers over a multivariate time series with at most 30 features, finding both intra and inter-packet correlations. A wide range of algorithms that can be used to detect anomalies in time series exist (a survey of outlier detection techniques can also be found in [26]), but they are mostly limited to continuous variables (we have discrete and categorical values) and to strictly ordered series. Packets are neither totally numeric nor strictly ordered. Missing these characteristic, we cannot use powerful mathematical instruments such as spectral analysis. Additionally, since in a real world situation it would be difficult to collect a large base of attack-free traffic in order to train the algorithm, we need it to be resistant to the presence of outliers in the training dataset.

Excluding supervised algorithms, we are left with a handful of candidates. A first approach (which we tested in [13]) is to map a time series onto a rolling window of observation, and then use regular clustering techniques for finding outliers. For instance, SOMs have been used in this fashion, [7], [27] on either connection data, or on packet header data (discarding the payload). However, [28] notes that this approach may be deeply flawed for statistical reasons.

PHAD, Packet Header Anomaly Detection [5], is a simple statistical modeling method which has been applied to data extracted by packet headers. By using a really simple method (which grants great performances), PHAD detects about half of the attacks in the DARPA 1999 dataset. The algorithm could be easily extended with the classification output by the first tier of our architecture. NETAD [9] is an evolution of PHAD and LERAD (Learning Rules for Anomaly Detection [8]). NETAD prefilters traffic using various rules (based on protocol type and sequence numbers), and then models nine non-disjunctive subsets of traffic. Both PHAD and NETAD have the conspicuous disadvantage that they do not identify intra-packet anomalies, but just inter-packet sequence anomalies. In addition, they require to be trained on attack-free datasets.

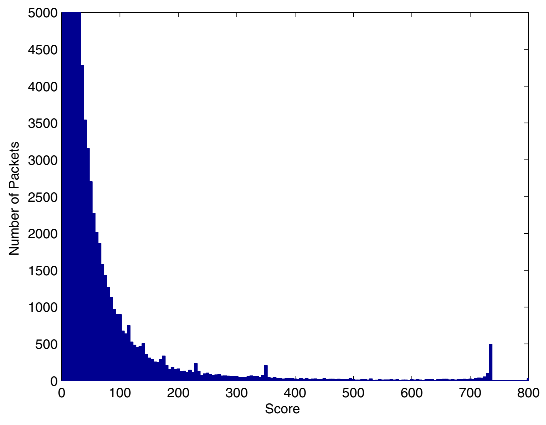

Fig. 2. Distribution of scores

In [29] a framework is proposed for using Hidden Markov Models for modeling multivariate time series. Information theoretic methods such as the Parzen Window method have been proposed in [6]. They have the advantage that being formulated as a statistical hypothesis test uses as a parameter an "acceptable false detection rate" which can be used for tuning, and that they do not need training. However we experimented their runtime to be unacceptable.

A much better alternative is to use a discounting learning algorithm such as SmartSifter [30]–[32]. It is designed for online usage, and it uses a "forgetting factor" in order to adapt the model to non-stationary data sources. The output of SmartSifter is a value expressing the statistical distance of the new observation, which means "how much" the new observation would modify the model currently learned. SmartSifter has also the great advantage to be able to use both categorical (SDLE algorithm) and metric variables (SDEM algorithm: in particular we used the parametric version).

We needed to modify SmartSifter to suit our needs. In particular, as the *Hellinger distance* required by the algorithm cannot be easily computed, we had to resort to heuristics and approximations (as pointed out in [31]). Also, in order to automatically tune the threshold beyond which a data vector is to be considered an outlier, we modified SmartSifter by introducing a training phase during which the distribution of the anomaly scores is approximated, and an estimated quantile of the distribution is also computed. In this way we can directly set the IDS sensitivity as the percentage of packets we want to consider as outliers.

As we can see from Figure 2, the distribution of scores cannot be approximated well by a normal distribution, so computing the sample mean and variance would not be of help. Therefore, we discretized the distribution with a quantization interval i, assuming that in each interval there is a uniform distribution, and computing the quantile of this discretized approximation. Of course, increasing i makes training faster and the approximation rougher. The quantile Q of order $q \in [0,1]$ can be determined through the following formula:

$$Q(q) = s \cdot \left(j + \frac{q - \sum_{i=0}^{j} P(S_H \in I_i)}{P(S_H \in I_{j+1})} \right),$$

where I_j is the interval in the discretization such that $P(S_H \in I_j) < q$ and $P(S_H \in I_{j+1}) \geq q$. Thus, setting the threshold of the anomaly score to $Q(q)$ the ratio of packets flagged as anomalous is more or less q.

As opposed to this totally unsupervised outlier determination, in [32] the authors of SmartSifter proposed a mixed supervised/unsupervised approach.

Feature selection is an important step for any learning application [33]. We wish to stress this point, since the algorithms proposed in the literature have been applied to more or less arbitrary selections of features of the packets. In an unsupervised setting such as ours, unsupervised techniques for feature selection should be employed [34]. Unsupervised FSS techniques usually compute the similarity between features and removes redundancies in order to reduce the number of features. Usually this is accomplished through partitioning or clustering of the original feature set into partitions, each of which will be then represented by a single representative feature to form the reduced subset. However, no reliable method exists which takes into account categorical variables.

Therefore, we resorted to a much simpler approach, by testing different combinations of the variables. As a first consideration, we chose to use only categorical variables, since the SDLE algorithm is much more efficient than the SDEM algorithm (which contains matrix inversions). A set containing source port, destination port, TCP flags, source and destination address and the payload classification worked best in our setup.

TABLE II
DETECTION RATES AND FALSE POSITIVE RATES FOR OUR PROTOTYPE

Threshold	Detection Rate	False Positive Rate
0.03%	66.7%	0.031 %
0.05%	72.2%	0.055 %
0.08%	77.8%	0.086%
0.09%	88.9%	0.095%

V. EXPERIMENTAL EVALUATION

In order to evaluate our architecture in a repeatable manner, we ran the prototype over various days of traffic drawn from the 4th week of the 1999 DARPA dataset [35]. We also added various attacks against the Apache web server and against the Samba service generated through the Metasploit framework (www.metasploit.org). The average results are reported in Table II. The first column contains the sensitivity threshold set for the algorithm, which is, as we noted, a good statistical predictor of the percentage of data that will be flagged as outliers by the algorithm. Therefore it is also a good predictor of the false positive rate, if the attack rate is not too high. The prototype is able to reach a 66.7% detection rate with as few as 0.03% false positives.

For comparison, the authors of SmartSifter in [31] tested their algorithm against the KDD Cup 1999 [36] dataset, which

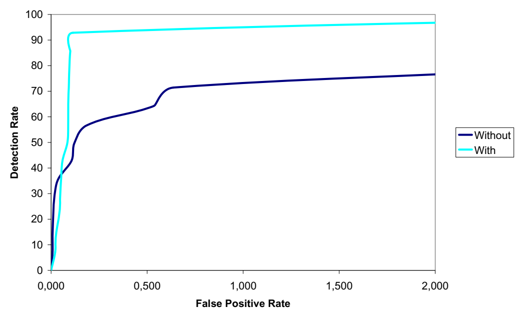

Fig. 3. ROC curves comparing the behavior of SmartSifter with (lighter) and without (darker) our architecture

is extracted from the same dataset we used by converting the tcpdump records in connection records by a traffic reconstruction tool. In the original test three continuous variables (duration, bytes transmitted from source, bytes transmitted from destination), and a categorical one (the service) were used. The authors of SmartSifter claim a 18% detection rate, with a 0.9% false positive rate (6421 connections). Our algorithm can instead reach a 92% detection rate with a 0.17% false positive rate (2035 packets), thus demonstrating a highly superior performance.

In Figure 3 we further show how our 2-tier architecture benefits the detection rate by comparing the ROC curves of the SmartSifter system with and without the payload classification tier by including and excluding the feature. The results are clearly superior when the first tier of unsupervised clustering is enabled, proving the usefulness of our approach.

PAYL [12] is a prototype of intrusion detection system which uses part of the payload of packets: in fact, it is the only instance in literature, besides our own work, where such a concept is applied. PAYL builds a set of models of payload $M_{i,j}$ depending on payload size i and destination port j. Each model $M_{i,j}$ contains the average frequency, and the standard deviation, of each of the 256 possible byte values. In the detection phase, the model M of each packet is computed, and compared against the model $M_{i,j}$ created during training using a roughly simplified form of the Mahalanobis distance (a distance measure for statistical distributions). In order to avoid an explosion in the number of models, during training models that are similar are aggregated. The merging step aggregates couples of models $M_{i,j}$ and $M_{k,j}$ if $k \simeq i$ and the Manhattan distance between the two models is below a threshold. PAYL does not take into account the header information. It also ignores the correlation over time in the packet flow. An additional limitation of PAYL is that it needs attack free traffic for training, even if the authors suggest, without demonstration, that if attacks are in minority with respect to normal packets.

Analyzing [12], the best overall results for PAYL show

TABLE III
DETECTION RATES AND FALSE POSITIVE RATES WITH HIGH FRAGMENTATION AND USE OF EVASION TECHNIQUES

Threshold	Detection Rate	False Positive Rate
0.05%	28.6%	0.07 %
0.08%	35.8%	0.09%
0.8%	57.1%	0.57%
1.79%	64.28%	1.32%

a detection rate of 58.7%, with a false positive rate that is between 0.1% and 1%. Our architecture can reach the same detection rate with a false positive rate below 0.03%, thus an order of magnitude better than PAYL, or on the other hand reach a 88.9% detection rate with no more than a 1% rate of false positives.

We also tested, in a very limited way, the resistance of the proposed architecture against fragmentation. In order to do this, we used *Fragroute* [37] to artificially introduce fragmentation in the DARPA dataset (in order to create fragmented, but normal packets and connections), fragmenting the 5% of the TCP traffic.

Then, we introduced fragmentation and evasion techniques on the attacks. Our results are shown in Table III. Of course, in such an extreme situation, the signal to noise ratio of the algorithm is worse, but still comparable or better than the results that the other systems have on the non-fragmented data of the IDEVAL dataset.

VI. Conclusions, Current Developments and Future Works

In this work we have summarized our research on the topic of unsupervised learning technologies and their application to the problem of intrusion detection, focusing in particular to the applications for network intrusion detection. After briefly analyzing the state of the art of the research in the field, we have described the challenges we met while implementing an innovative model of anomaly based network intrusion detection system, completely based on unsupervised learning techniques. We have described a novel, two tier architecture for such a system. We have shown how a first tier of clustering (based on Self Organizing Maps) can perform an efficient, unsupervised pattern recognition on packet payloads. We have considered possible alternate metrics for clustering, and shown how the euclidean metric performs overall. We have also shown how the curse of dimensionality requires an appropriate resolution, and proposed various heuristics to improve the runtime efficiency of the algorithm, obtaining a throughput rate almost three times higher than the original one, with marginal misclassification rates, without truncating the number of the bytes of the payload taken into account.

We have described how we combined this first tier with a modified version of the SmartSifter outlier detection algorithm, and we have given results on the detection rate and false positive rate, showing that the system outperforms a similar, state-of-the-art system by almost an order of magnitude in term of false positive reduction. We have also studied how the errors introduced by our heuristics affect the algorithm detection capabilities, and concluded that our modified algorithm works as well as the original version of the SOM.

Future works on this system will strive to further improve its speed, as well as to reduce the false positive rate as much as possible. A theme we are beginning to research on now, and which is the natural evolution of this work, is how to integrate the network and host-based systems we designed, in order to use the results of both to automatically filter out false positives and to improve correlation and alert quality.

ACKNOWLEDGMENTS

Most of the work was supported in part by the Italian FIRB Project "Performance evaluation for complex systems", and in part by a PhD scholarship of the Italian Ministry for the University and Research (MIUR). We need to thank prof. Sergio M. Savaresi, prof. Salvatore J. Stolfo and Dr. Giuliano Casale for their precious suggestions for improvement over our previous work. We also need to thank warmly our students Matteo F. Zazzetta and Federico Maggi for their precious support in software development and lab testing.

REFERENCES

[1] J. P. Anderson, "Computer security threat monitoring and surveillance," J. P. Anderson Co., Ft. Washington, Pennsylvania, Tech. Rep., Apr 1980.

[2] R. G. Bace, *Intrusion detection*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc., 2000.

[3] T. H. Ptacek and T. N. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," Secure Networks, Calgary, Canada, Tech. Rep. T2R-0Y6, 1998. [Online]. Available: citeseer.nj.nec.com/ptacek98insertion.html

[4] S. Zanero, "Unsupervised learning algorithms for intrusion detection," Ph.D. dissertation, Politecnico di Milano T.U., May 2006. [Online]. Available: http://home.dei.polimi.it/zanero/papers/tesi_zanero_online.pdf

[5] M. Mahoney and P. Chan, "Detecting novel attacks by identifying anomalous network packet headers," Florida Institute of Technology, Tech. Rep. CS-2001-2, 2001.

[6] D.-Y. Yeung and C. Chow, "Parzen-Window network intrusion detectors," in *Proc. of the 16th Int'l Conf. on Pattern Recognition*, vol. 4, aug 2002, pp. 385–388. [Online]. Available: citeseer.nj.nec.com/505644.html

[7] K. Labib and R. Vemuri, "NSOM: A real-time network-based intrusion detection system using self-organizing maps," Dept. of Applied Science, University of California, Davis, Tech. Rep., 2002.

[8] M. V. Mahoney and P. K. Chan, "A machine learning approach to detecting attacks by identifying anomalies in network traffic," Florida Institute of Technology, Tech. Rep. CS-2002-08, 2002.

[9] M. V. Mahoney, "Network traffic anomaly detection based on packet bytes," in *Proceedings of the 19th Annual ACM Symposium on Applied Computing*, 2003.

[10] M. V. Mahoney and P. K. Chan, "Learning rules for anomaly detection of hostile network traffic," in *Proc. of the 3rd IEEE Int'l Conf. on Data Mining*, 2003, p. 601.

[11] ——, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2002, pp. 376–385.

[12] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *RAID Symposium*, September 2004.

[13] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proc. of the 2004 ACM Symposium on Applied Computing*. ACM Press, 2004, pp. 412–419.

[14] J. Han and M. Kamber, *Data Mining: concepts and techniques*. Morgan-Kauffman, 2000.

[15] S. Zanero, "Analyzing tcp traffic patterns using self organizing maps," ser. Lecture Notes in Computer Science, F. Roli and S. Vitulano, Eds., vol. 3617. Cagliari, Italy: Springer, September 2005, pp. 83–90.

[16] D. Hawkins, *Identification of Outliers*. London: Chapman and Hall, 1980.

[17] D. Boley, V. Borst, and M. Gini, "An unsupervised clustering tool for unstructured data," in *IJCAI 99 Int'l Joint Conf. on Artificial Intelligence*, Stockholm, Aug 1999.

[18] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin: Springer-Verlag, 2001.

[19] I. T. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[20] S. Zanero, "Improving self organizing map performance for network intrusion detection," in *SDM 2005 Workshop on "Clustering High Dimensional Data and its Applications"*, 2005.

[21] S. McCreary and K. Claffy, "Trends in wide area ip traffic patterns - a view from ames internet exchange," in *Proc. of ITC'2000*, 2000.

[22] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" *Lecture Notes in Computer Science*, vol. 1540, pp. 217–235, 1999. [Online]. Available: citeseer.ist.psu.edu/beyer99when.html

[23] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" in *The VLDB Journal*, 2000, pp. 506–515. [Online]. Available: citeseer.ist.psu.edu/hinneburg00what.html

[24] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," *Lecture Notes in Computer Science*, vol. 1973, 2001. [Online]. Available: citeseer.ist.psu.edu/aggarwal01surprising.html

[25] C. C. Aggarwal, "On effective classification of strings with wavelets," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2002, pp. 163–172.

[26] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, 2004.

[27] P. Lichodzijewski, A. Zincir-Heywood, and M. Heywood, "Dynamic intrusion detection using self organizing maps," in *14th Annual Canadian Information Technology Security Symp.*, May 2002.

[28] J. Lin, E. Keogh, and W. Truppel, "Clustering of streaming time series is meaningless," in *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. New York, NY, USA: ACM Press, 2003, pp. 56–65.

[29] S. Kirshner, "Modeling of multivariate time series using hidden markov models," Ph.D. dissertation, Department of Computer Science, University of California, Irvine, March 2005.

[30] K. Yamanishi, J. ichi Takeuchi, G. J. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," in *Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Aug 2000, pp. 320–324. [Online]. Available: citeseer.nj.nec.com/yamanishi00line.html

[31] K. Yamanishi, J.-I. Takeuchi, G. J. Williams, and P. Milne, "Online unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Knowledge Discovery and Data Mining*, vol. 8, no. 3, pp. 275–300, 2004.

[32] K. Yamanishi and J. ichi Takeuchi, "Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2001, pp. 389–394.

[33] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[34] S. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining: Scalability, Knowledge Discovery, and Soft Granular Computing*. Boca Raton, FL: Chapman Hall/CRC Press, May 2004.

[35] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 DARPA off-line intrusion detection evaluation," in *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*. London, UK: Springer-Verlag, 2000, pp. 162–182.

[36] "KDD Cup '99 Dataset," available online at http://kdd.ics.uci.edu/databases/kddcup99/kddcu99.html.

[37] "Fragroute," available online at URL http://www.monkey.org/~dugsong/fragroute/.