Unsupervised learning techniques for an intrusion detection system

Stefano Zanero
zanero@elet.polimi.it

Sergio M. Savaresi
savaresi@elet.polimi.it

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. da Vinci, 32; 20133, Milan, Italy

ABSTRACT

With the continuous evolution of the types of attacks against computer networks, traditional intrusion detection systems, based on pattern matching and static signatures, are increasingly limited by their need of an up-to-date and comprehensive knowledge base. Data mining techniques have been successfully applied in host-based intrusion detection. Applying data mining techniques on raw network data, however, is made difficult by the sheer size of the input; this is usually avoided by discarding the network packet contents.

In this paper, we introduce a two-tier architecture to overcome this problem: the first tier is an unsupervised clustering algorithm which reduces the network packets payload to a tractable size. The second tier is a traditional anomaly detection algorithm, whose efficiency is improved by the availability of data on the packet payload content.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access (e.g., hacking, phreaking); I.5.3 [**Clustering**]: Algorithms; C.2.3 [**Network Operations**]: Network monitoring

General Terms

Security, Experimentation.

Keywords

Intrusion detection, anomaly detection, unsupervised clustering, quality of clusters, K-means, principal direction divisive partitioning, self-organizing maps.

1. INTRODUCTION AND MOTIVATIONS

One of the most excruciating pains in both the intrusion and virus detection fields is the constant need for up-to-date definition of the attacks. This follows from the use of a "misuse detection" approach, which tries to define what is anomalous instead of defining what is normal. While this kind of approach has been widely successful and is implemented in almost all the modern antivirus and intrusion detection tools, its main drawback is that, when facing an unknown attack, misuse-based systems are substantially useless. In the antivirus world this problem has been more or less successfully approached with round-the-clock response team and signature distribution methodologies. In the intrusion detection world, maintaining such a knowledge base up to date is substantially a lost battle.

The problem does not lie only in the sheer number of vulnerabilities that are discovered every day: there is also an unknown number of unexposed vulnerabilities that may not be immediately available to the experts for analysis and inclusion in the knowledge base (which, in general, does not happen for viral code). In addition, some forms of attacks could even be studied by a particularly skilled attacker on the spot, just to hit a single or a few systems (again, this is not what you would expect from a virus). In fact, misuse-based IDS are particularly effective against the so-called "script kiddies", unskilled attackers that rely on commonly known attack tools, for which a signature is usually widely available.

Additionally, computer attacks are usually polymorph, since there are different ways to exploit the same vulnerability. Thus, it is correspondingly more difficult to develop appropriate signatures: either we generate a number of signatures to cover each possible variation of the attack, or we try to generalize the signatures, risking to generate false positives.

In some cases this is inherent to the attacks, for instance the "unicode" related bugs, since for each character there are multiple possible Unicode encodings. But let us examine the ADMutate tool (http://www.ktwo.ca/c/ADMmutate-0.8.4.tar.gz), developed by the Canadian hacker "K2": this tool enables an aggressor to encrypt the shellcode of a stack-smashing buffer overflow attack [21], and to append this encrypted shellcode to the decryption algorithm. Even if now most IDS have a specific signature for the decryption code specific to ADMutate it's easy to understand that this principle can be indefinitely applied, in many forms.

An obvious solution would be to go back to the basics, and try to implement an anomaly detection approach, modeling what is *normal* instead than what is *anomalous*. This is surprisingly similar to the earliest conceptions of what an IDS should do [1].

Surprisingly enough, anomaly detection systems have been

SAC'04 March 14-17 2004, Nicosia, Cyprus

successfully implemented (at least in academic projects) in an host-based fashion, but have so far spectacularly failed to be useful in network-based systems, with a few exceptions. This is due both to commercial reasons, and to real problems with anomaly detection systems. They show an alarming tendency to generate huge volumes of false positives, they do not clearly define "what is wrong", but instead rely on statistical measures of "weirdness". In addition, it has always been a difficult task for researchers to understand what to monitor in a network traffic flow, and how to describe and model it. Specific, limited "anomaly detection" signatures have been developed and implemented in traditional commercial IDS systems, in order to detect common signs of attacks (for instance, the presence of binary codes in unusual places). However, there are no (or very few) fully fledged network-based anomaly detection systems.

In this work we propose a novel architecture for a network-based Intrusion Detection System based on unsupervised learning and data mining techniques. We believe research in this particular field to be increasingly important, since the misuse detection approach has been widely studied and implemented, but is now beginning to show its limits; so, it is our opinion that future research should explore anomaly detection systems to properly complement existing misuse detection ones.

It is important to note that, since failures and strengths of such approaches are symmetric, some systems try to integrate different approaches [26], but there are difficult and intriguing problems of metrics, fusion and normalization when working on data coming from different sources, somehow tied to the "multisensor data fusion" problems already under consideration in the field of robotics [2]. We will not try to address such problems, due to space limits; however, it is important to keep in mind that we are trying to develop a paradigm that should complement, not substitute entirely, the misuse detection approach.

Our proposed architecture aims to be efficient and properly structured to be realistically implemented: to this end, we have studied carefully the performance of the algorithms. We also made use of real network data, not generated ad hoc by us for test purposes, to ensure that our results are reliable and not biased by our own unconscious assumptions.

The remainder of the paper is organized as follows: in section 2 we will propose a novel architecture for a Network IDS completely based on unsupervised learning techniques. We will examine both the tiers of the proposed architecture in the following sections; in particular, section 3 will focus on the results we obtained in the use of clustering algorithms on the payload of TCP packets, and section 4 will introduce the concept of anomaly detection on sequences of data and proposes our preliminary results on the subject. Finally, in section 5 we will draw our conclusions, and set the course for further developments.

2. A TWO-TIER ARCHITECTURE

We propose an innovative architecture for a network-based anomaly detection IDS, based on unsupervised learning algorithms. We chose to focus on this class of algorithms because they exhibit the following interesting properties:

- Outlier detection: unsupervised learning techniques are capable of identifying "strange" observations in a wide range of phenomena; this is a characteristic we definitely need in an anomaly based IDS;
- Generalization: unsupervised learning techniques are also quite robust and gave us the hope of being able to resist to polymorphic attacks;
- Unsupervised learning: we wanted to create a model totally orthogonal to the misuse based model, which is dependent on the input of expert knowledge, so we tried to develop an IDS which needed no a priori knowledge inputs;
- Adaptation: a learning algorithm can be tuned totally to the specific network it operates into, which is also an important feature to reduce the number of false positives and optimize the detection rate.

The problem we are trying to analyze in the unsupervised learning framework is the following: *we wish to detect anomalies in the flow of packets on a TCP/IP network.* This apparently simple problem statement hides a number of subproblems.

A TCP/IP packet, over an Ethernet network, has a variable dimension between 20 and 1500 bytes. The first 20 bytes (or more, depending on the number of options) constitute the IP header, and we possess full knowledge of its meaning. Another sequence of up to 20 bytes is the header of the upper layer protocol, usually either TCP, UDP or ICMP. We have full knowledge also of the meaning of this header, but in the case of connection-oriented protocols (most notably TCP) these headers may need inter-correlation in order to be fully deciphered.

If we look at the data carried by the packet (the payload), the situation grows even more complex: we could add knowledge about upper layer protocols such as HTTP, FTP and so on, but this would add complexity, decrease the generality of the IDS (limiting it to well-known protocols) and generally imposing a performance cost. It is computationally infeasible to perform a full, real-time reconstruction of the sessions of traffic of a large network at such a high level. In addition, the higher the protocol layer we consider, the more the IDS will become sensible to reconstruction problems, possibly leading to attack windows [27].

However, even if we do not try to reconstruct sessions, we cannot hope to understand what is going on by looking at one packet at a time: we need to correlate what is happening over time, either by observing a rolling window of packets in parallel or by using algorithms with a memory property.

Unfortunately, the computational complexity of unsupervised learning algorithms scales up steeply with the size of the considered data. A realistic (even ambitious) limit is of a thousand, or little more, dimensions (or *features*). So, if we tried to apply an unsupervised learning algorithm directly to raw data, we could feed the algorithm only with a single packet at a time. As we said before, this would greatly impair the ability of the algorithm to discover correlation between packets over time.

A few algorithms can be optimized to treat data with many thousands of dimensions, but only in the case that they are sparse (for instance, a word/document incidence matrix in a document classification and retrieval problem [5]), but we are dealing with dense data. Our tests have also shown that traditional dimension-reduction techniques, such as dimension scaling algorithms [4] or Principal Component

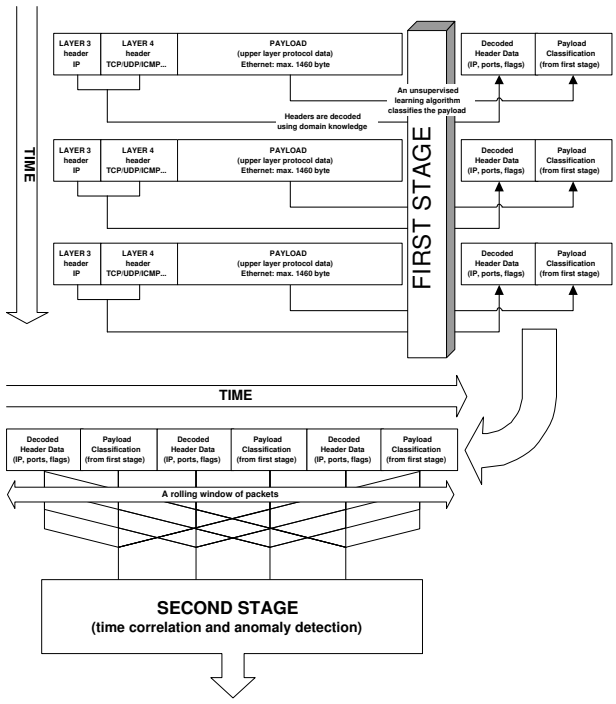

Figure 1: Architecture of the IDS

Analysis [13], are quite ineffective to treat these data, since by their nature they tend to "compress" outliers, and this is exactly the opposite of what we want to achieve.

Many existing researches on the use of unsupervised learning algorithms for intrusion detection purposes solve this problem by discarding the payload and retaining only the information in the packet header. This is clearly not an optimal solution, since it leads to an unacceptable information loss: most attacks, in fact, are detectable only by analyzing the payload of a packet, not the headers alone. The algorithms show nevertheless interesting, albeit obviously limited, intrusion detection properties. In section 4 we will analyze them in depth, with their points of strength and their shortcomings.

In order to solve this problem, we developed the concept of a two-tier intrusion detection system, which allows us to retain at least part of the information related to the payload content. Our work hypothesis was that on most networks, the traffic would belong to a small number of services and protocols, regularly used, and so that most of it would belong to a relatively small number of classes.

Thus, in the first tier of the system, an unsupervised clustering algorithm classifies the payload of the packets, observing one packet at a time and "compressing" it into a single byte of information. This classification can be added to the information decoded from the packet header (or 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 clustering.

The architecture is fully explained in Figure 1. We have analyzed both the tiers and built functional prototypes that have confirmed our hypotheses.

3. CLUSTERING THE PAYLOAD

3.1 Requirements and algorithm selection

In the first tier of our architecture we need to find an algorithm that receives in input the payload of a TCP or UDP over IP packet (on an ethernet segment this means up to 1460 byte values which can be interpreted as vectors of variable size) and classifies them in a "sensible" way. An ICMP packet does not pose problems, since the simple decoding of the protocol header usually gives us all the information we need (albeit some covert communication protocols use the payload of an ICMP packet this can be ignored for simplicity at this level of analysis).

TCP is used to carry the data of an high number of upper layer protocols. We do not want, however, to rely on domain knowledge (for example the well-known ports list) to predivide traffic by type, for at least three main reasons. First, if we use the port field (or any other indicator in the header) as a trusted information source, we would be assuming that every port is actually used for its proper communication protocol. Thus a connection which for example gives the output of a shell command over an HTTP communication channel would not be detected an anomalous connection.

Secondly, previous researches have shown that neural algorithms can recognize protocols automatically [31], so a clustering algorithm should be able to do the same. We will show that this is true, at least partially.

Finally, we set up to use as little a priori knowledge as possible in our research, trying to understand how much knowledge could be gathered by using unsupervised algorithms alone. In [20] the authors fully explore the complementary approach of using supervised data mining techniques along with domain knowledge for intrusion detection purposes, with interesting results.

A "sensible" automatically generated classification should, in principle, keep as much information as possible for the second tier algorithm about the "similarity" between packets. Obviously, since our end goal is to detect intrusions, the classification should also show the property to separate packets with anomalous or malformed payload from normal packets, and should also divide the payloads reflecting the divisions between protocols as well as possible.

This is a typical clustering problem. A classic definition of clustering is:

Definition 1. Clustering is the grouping of similar objects from a given set of inputs [10].

Another is:

Definition 2. An algorithm by which objects are grouped in classes, so that intra-class similarity is maximized and inter-class similarity is minimized [9].

Even if conceptually simple, there is an endless variety of algorithms designed to solve this problem. To solve a clustering problem we must decide both the measure of similarity between elements, and an efficient algorithm to find an acceptable solution, since finding the "optimal" solution (which maximizes both the intra-class similarity and minimizing inter-class similarity) would be an NP-hard problem.

For many algorithms we also need a criterion to define a correct or acceptable number of classes, while some algorithms automatically discover this number and others are quite tolerant to an arbitrarily high choice.

Another interesting characteristic of many clustering algorithms is a built-in capability to detect outliers, classically defined as follows:

Definition 3. An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism. [11]

We studied most of the proposed algorithms (we cannot report our findings in this paper, but a comprehensive review can be found in [12]), and we chose to implement three widely used techniques, representative of three different approaches to the problem: the K-means algorithm, which is a centroid-based approach; the principal direction partitioning, a hierarchical divisive approach [3]; and Kohonen's Self Organizing Maps algorithm [15], which is a competitive neural approach.

3.2 Experiments and results

As noted before, we needed experimental data to feed into the algorithms, in particular dumps of common network traffic, in the format described by the "libpcap" libraries. As of our knowledge, there is only one source of test data which makes the full payload available for inspection: the dataset created by the Lincoln Laboratory at M.I.T., also known as "DARPA IDS Evaluation dataset". In particular, for the experiments described in this paper, portions of the 1998 dataset, which are commented and described by a master's thesis [14].

It is important to note, however, that these data have been artificially generated specifically for IDS evaluation. In fact, in [25] there is a detailed analysis of the shortcomings of this traffic sample set. In particular, the author notes that no detail is available on the generation methods, that there is no evidence that the traffic is actually realistic, and that spurious packets, so common on the Internet today, are not taken into account. While this is probably true, it does not actually matter a lot for our experiments. We need to be able to show that our architecture is capable of detecting attacks mixed in background data, and we can do this only under test condition.

As a preliminary step, we developed a parser to import data from the libpcap format into the Matlab environment, since it seems that there are no tools currently available to do this. This tool will be released under GPL. Our experimental implementation was limited for simplicity to the TCP protocol over IP, but an extension to the UDP protocol would be straightforward.

None of the algorithms we chose predefines a metric for distance, and as a tentative experiment we chose to use a simple euclidean distance criterion between vectors of data. We are conscious that this choice has no real theoretical support, but our experiments have shown that it works well. More work could be done to study other, maybe better suited, distance functions. In particular, we are currently studying lexical distances as an alternative.

We trained the algorithms with a wide variety of parameters, on various subsets of the dataset. It is important to discuss how we evaluated the results. There are three main ways to evaluate an unsupervised classification:

- Inspection-based: by manually inspecting the classification and checking if it "makes sense" to us;
- Expert-based: by letting an expert manually classify the same data and see if the results are comparable;
- Task-based: by evaluating the algorithm against the result of the task it is trying to accomplish; in our case this would mean evaluating it using the performances of the complete architecture as a criterion.

While the latter method is the most appealing, we needed some preliminary criteria to evaluate the algorithms even without a fully functional architecture. We used the expert classification for little datasets, but it was impossible to apply to huge volumes of payloads; so we resorted to both manual inspection and "proof of concept" tasks to evaluate the correctness of the classification.

In particular, in Figure 2 we present the results of the training of a 10×10 Self Organizing Map that creates a division of the data in 100 clusters. The network was trained for 10.000 epochs. The histograms represent the number of packets (on y-axis) present in each cluster (on x-axis). Please remind that for graphical reasons the number of packets on y-axis may be differently scaled in the various pictures.

The network was trained with a representative subset of traffic, and then used to classify two sets of about 2000 packets: the first representing normal (albeit different) traffic, the second being the dump of a vulnerability scan with the "Nessus" tool (www.nessus.org). We used a Nessus scan as representative of anomalous traffic, since it is a really brutal aggression which should not go unnoticed, and generates also a huge volume of TCP traffic. We delayed experiments with true network attacks for the second stage tests (see below at section 4). The difference in the distribution of packets is noticeable, and this is a first proof of the usefulness of our approach (albeit not complete!). In the next section we will show how such differences over a time window can be detected, and we will also show that the classification results are significative even on a packet-by-packet base.

Additionally, manual inspection proved that most of the resulting clusters made "sense", which means that the packets falling in the same classes were either the same type of files, or the same portions of protocols (i.e. all the e-mail traffic fell into a narrow group of classes; all the FTP commands fell into another group of classes ...).

We noted an extreme inefficiency in Matlab built-in algorithms. A network training time is about linear in the product of the map dimensions (i.e. for a $n \cdot m$ network, the time is about $O(n \cdot m)$). The training time is linear in the number of epoch, $O(n)$. But, strangely, in the Neural Network Toolbox implementation of Kohonen's algorithms the training time is not linear in the number of items in the training set, nor in the number of dimensions of the vectors: it grows linearly in both dimension and cardinality until it exhausts system resources, but afterwards the I/O costs make it explode exponentially. The S.O.M. Toolbox (http://www.cis.hut.fi/projects/somtoolbox/) showed a much better behavior.

In Figure 3 we present instead the results of a division in 50 classes operated by the principal direction divisive partitioning algorithm, in the same experimental conditions used for the S.O.M. traffics. We can see that also in this

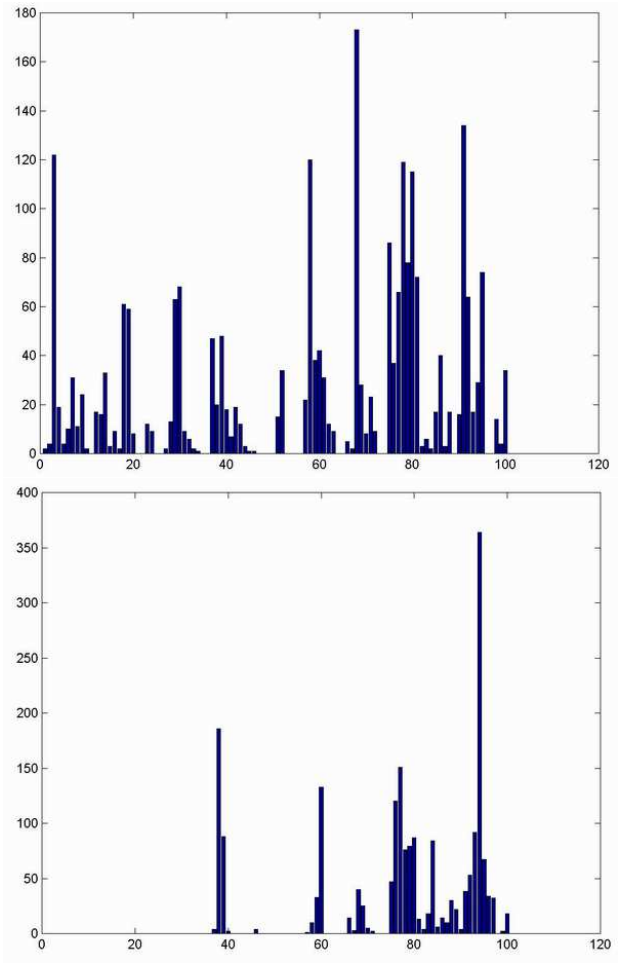

Figure 2: Comparison between the classification of normal traffic (above) and Nessus traffic (below) by a 10x10 S.O.M. network.

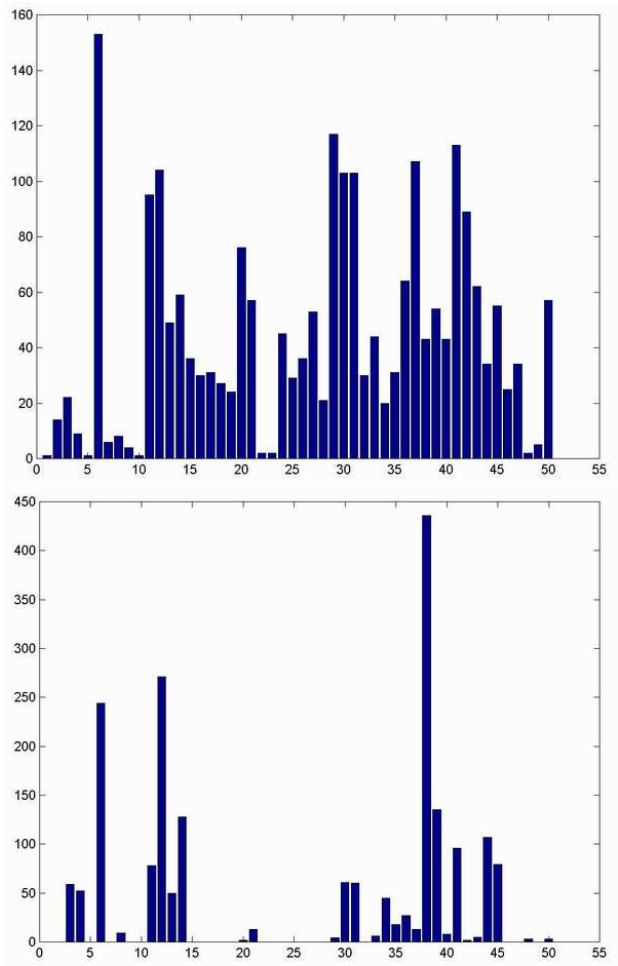

Figure 3: Comparison between the classification of normal traffic (above) and Nessus traffic (below) over 50 classes by a principal direction algorithm

case the distribution of packets varies wildly between normal and Nessus traffic. The manual inspection also confirms the impression of a sensible classification.

It is well worth noting that in the PDDP algorithm there is an additional open problem to solve. At each step of the algorithm we must choose the cluster which is going to be split. We would like, obviously, to choose the most "scattered" leaf of the hierarchical divisive tree. Various ways to define the scattering of a leaf have been studied in [29], but for our implementation we chose the simplest (a measure of variance). Other variants could certainly be experimented, and maybe lead to better results.

The computational cost of the PDDP algorithm during training is critical, because it happens that the first step of the algorithm is the most costly (since the training set is split at each step). Normally, for computing the Principal Direction, Matlab uses a SVD (Singular Value Decomposition) algorithm with a time complexity $O(p \cdot q^2 + p^2 \cdot q + q^3)$ with p and q sizes of the matrix (which means cardinality and dimensionality of the training set). This proved impractical, so we used an highly efficient implementation of the Lanczos algorithm, with bidiagonalization and partial reorthogonalization, which offers a complexity of $O(p \cdot q \cdot r2)$, where r is the rank of the matrix and so $r = min\{p, q\}$ [18]. However, even this algorithm shows difficulties when the cardinality of the training set grows. The problem could be difficult to solve in a real world application.

In Figure 4, finally, we see that the K-means algorithm does not behave as well as the other two algorithms. Aside from the distribution of traffic which is not as distinct, manual inspection reveals that K-means clusters are less respondent to our expectations. In addition, the random initialization of the algorithm makes the quality of the final result unpredictable, since it converges rapidly to a local (not global) minimum in the distribution of the centroids.

K-means is globally the fastest algorithm we tested, showing really no performance problem; however, the corrections necessary to eliminate or reduce the random initialization weakness (for instance using the so-called "global K-means" algorithm [23]) make the algorithm completely untractable. There is, however, a divisive variant of the K-means algorithm, which is compared to the PDDP algorithm in [28] and for which interesting properties hold, which could solve the locality problem. We will repeat our tests on it as our research proceeds.

Overall, it is our opinion that the S.O.M. algorithm works

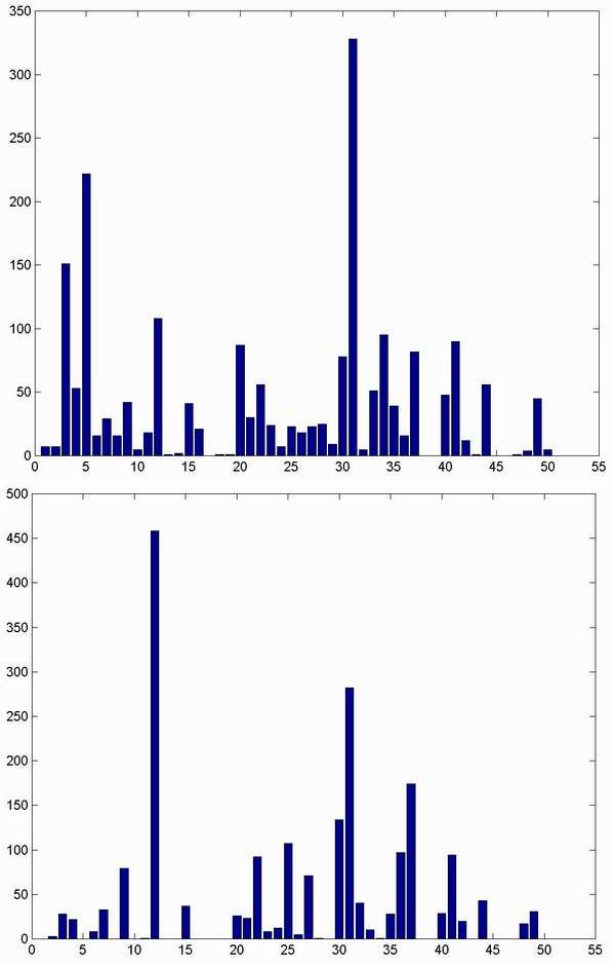

Figure 4: Comparison between the classification of normal traffic (above) and Nessus traffic (below) over 50 classes by a K-means algorithm

best, closely followed by the PDDP algorithm which is hampered by its performance problems. K-means could be a good choice if the locality problem was solved without unreasonable computational costs.

In a famous article, some years ago, J. Frank, [6], while commenting on the future trends of artificial intelligence, pointed at the clustering algorithms as a possible future approach to intrusion detection. Today we can overall confirm that his intuition was correct.

4. DETECTION OF ANOMALIES

4.1 Requirements and previous researches

Aside from clustering the payload, the first tier of our architecture also decodes the IP and TCP packet headers. The second tier receives in input a handful of data, about 30 entries, for each packet. This algorithm should mainly deal with two problems:

1. Intra-packet correlation: to analyze the content of each packet looking for indicators of anomaly;

2. Inter-packet correlation over time: to recognize anomalous distributions of packets (we saw a clamorous example in figures 2, 3 and 4).

To perform the second type of analysis, the algorithm must either observe a rolling window of data and/or be able to keep memory of past observations. There is an open trade-off at this point, since enlarging the time window (or, correspondingly, increasing the weight of the memory) for a better correlation could blind the system to atomic attacks, which represent a significative share of network-based attacks. In fact, most misuse-based network IDS work very well by analyzing a single packet at a time. But a statistical system, by its own nature, could be better at detecting significative variations over a long time than single attack packets.

There is a wide range of algorithms that can be used to detect anomalies in time series, but they are mostly limited to numeric and strictly ordered time series. Packets are neither totally numeric nor strictly ordered. Missing these characteristic, we cannot apply powerful mathematical instruments such as spectral analysis.

Excluding supervised algorithms and proposed algorithms lacking any real experimentation, there are just a handful of analysis methods that could be applied to our problem. Many of them have already been used for IDS purposes, mostly in host-based approaches, often examining the sequence of commands executed by a user [19]. The use of our two-tier approach which reduces network traffic to a few significative values could allow to apply such algorithms.

Instance Based Learning (IBL) is a class of algorithms which represent concepts by the means of a dictionary of "already seen" instances. There are both supervised and unsupervised variants, and an unsupervised one has been proposed for host-based intrusion detection purposes [17]. However, it seems that this algorithm works well for problems where the number of instances in the dictionary is quite limited. More studies would be needed to apply this approach to network data.

Clustering Algorithms are not directly suitable for this purpose, since they are not time dependent. However, they can be intuitively used for detecting time sequence anomalies by applying them to a rolling window of packets: some authors proposed the use of a S.O.M. to detect attacks in the DARPA dataset, by applying it not to packet data but to a TCP connection summary, with 6 characteristics for each connection [22]; others, instead, used a S.O.M. to analyze network traffic, discarding the payload and putting the header information in a rolling window. The prototype, called NSOM, can detect denial of service attacks [16]. We will see how their approach can be expanded and used as a suitable second stage in our architecture. Other authors propose to explicitly use time as a feature, and to show the packets to a S.O.M. one at a time [7]. This is, in our opinion, a mistake, since time on a network is even more relative than elsewhere, and even if a time scale could be derived, a S.O.M. cannot really handle a linearly increasing measure like that in a useful way.

P.H.A.D., Packet Header Anomaly Detection [24], is a simple statistical method, with high performances, which detects about 50% of the attacks in the DARPA '99 dataset. It could be combined with our first stage, but has the conspicuous disadvantage that it does not identify intra-packet anomalies, but just inter-packet sequence anomalies.

Information theoretic methods have also been proposed,

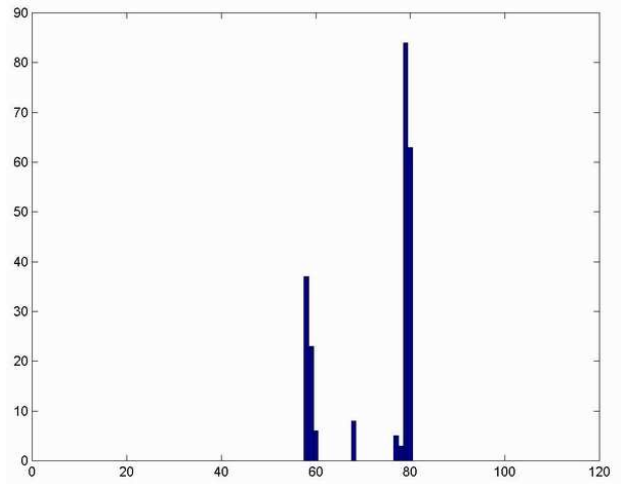

Figure 5: Classification of normal traffic on port 21/TCP by a S.O.M. algorithm

such as the Parzen Window method presented in [33]. Being formulated as a statistical hypothesis test, it has an "acceptable false detection rate" parameter that can be used for tuning; in addition, it does not need training, but it has an unacceptable running time. A discounting learning algorithm has been used in the Smart Sifter prototype [32], and should combine a smooth running time with good detection rates. However, it lacks reports of real world experimentation.

Other algorithms have been proposed to detect anomalies in a less general fashion, for instance various researchers have proposed algorithms to detect the spread of active worms on the base of the anomalous traffic they generate [30]. Albeit interesting, these algorithms lack the generality we are looking for.

4.2 Proposals and experiments

Our objective is to add the classification of payloads produced by the first stage as one of the features analyzed by any of these algorithms, thus extending their ability to detect attacks also to the content of the packet. A precondition is that attack payloads are "classified differently" from normal payloads. We can give proof of concept examples of this.

In Figure 5 we can see how a trained first stage 10×10 S.O.M. network classifies 5000 packets with the destination port set to 21/TCP (FTP service command channel). It can be observed how all the packets fall in a narrow group of classes. Any kind of unsupervised learning mechanism should be able to "learn" such a strong correlation.

Using the vulnerability database ArachNIDS (available at the URL: www.whitehats.com/ids), we ran the same S.O.M. on the packet captures of some FTP server attacks. For instance, the payload of the format string wu-ftpd bug exploit (ArachNIDS code IDS453) is classified in class 69, which is not one of the usual classes for FTP traffic.

Even more interesting is the result we obtain when analyzing the globbing denial-of-service attack (ArachNIDS code IDS487). The aggressor tries to overload the FTP server by sending a command similar to the following ones: `LIST */../*/../*/...`, or `LIST */.*/*/.*/*/...`, or any long combination of wildcards. The S.O.M. classifies all the known variations of the attack in a single class, number 97, which does not contain any normal FTP traffic. Signatures for this attack for misuse based IDS are difficult to write: this is an example of polymorphism in the attacks. We verify that S.O.M. algorithms are indeed resilient to these manipulations, while in order to achieve a generalized match with a signature based system we would need to write a signature matching `/*` (which is what Snort actually does), but this leads invariably to a number of false positives.

A buffer overflow attack (ArachNIDS IDS 287) is classified into class 81, which does not contain any normal FTP traffic. This attack uses the NOP Intel x86 hexadecimal code (`0x90`) as "padding", because it is sometimes difficult to understand where, exactly, code execution will begin. This is a common feature for many buffer overflow exploits: most IDS thus detect a long sequence of NOP as a possible shellcode. Sneaky attackers, however, use a jump to the following instruction (`0xeb 0x00`) instead of a NOP to fool the IDS. But even if we substitute the NOP codes with `0xeb 0x00` and run the algorithms again, the attack is still classified into class 81.

We also implemented a preliminary second-tier algorithm, using a S.O.M. to analyze a rolling window of packets. We employed different combination of parameters, but a 15×15 network used to analyze 10-20 packets at a time with a reduced feature set (including the source and destination ports, the TCP flags, and some of the IP header fields) gave us the best results. 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: our tests suggest that a deeper analysis should be done to guess which features are really important and which can be safely discarded. The importance of correctly choosing features for machine learning problem has been widely discussed in literature (see [8]), and our problem is not an exception: in our tests, keeping all the features effectively blinded the system, while selecting accurately a subset of them brought forth good results.

This experimental second tier has been applied to the network traffic both including and excluding the results of the first tier (in fact by excluding it, it substantially replicated the NSOM prototype [16]). Preliminary results show an increment of up to the 75 % in the detection rate over a sequence of attacks, but also (obviously) a limited increase in false positives. The system, with an appropriate buffer, can handle a 10 Mbps dataflow even in our test implementation. It could easily scale up to fast ethernet speeds.

On the basis of these observations, we can infer that complementing any of the algorithms described in 4.1 with a first stage of payload classification, as we described in this paper, can successfully increase the detection rate of the systems.

5. CONCLUSIONS

We have described an innovative model of Anomaly Based Network Intrusion Detection System, totally based on unsupervised learning techniques. We have described the overall architecture of the system and proposed our first empirical results on a test implementation. We have extensively tested candidate algorithms for the first tier, and albeit there is an open question on which kind of distance would better suit the payload contents. We have given proofs of the concept of the second tier, and also preliminary results on a proto-

type. These results, however, should be considered qualitative until the full architecture of the system is integrated and tested, and appropriate thresholds are studied in order to optimize the detection rate versus the false positive rate.

6. REFERENCES

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

[2] T. Bass. Intrusion detection systems and multisensor data fusion. *Comm. of the ACM*, 43(4):99–105, 2000.

[3] 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.

[4] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1995.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[6] J. Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proc. of the 17th Nat'l Computer Security Conf.*, Baltimore, MD, 1994.

[7] L. Girardin. An eye on network intruder-administrator shootouts. In *Proc. of the Workshop on Intrusion Detection and Network Monitoring*, pages 19–28, Berkeley, CA, USA, 1999. USENIX Association.

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

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

[10] J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

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

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surv.*, 31(3):264–323, 1999.

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

[14] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachussets Institute of Technology, 1998.

[15] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berling, 3 edition, 2001.

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

[17] T. Lane and C. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Trans. on Information and System Security*, 2(3):295–331, 1999.

[18] R. Larsen. *Lanczos bidiagonalization with partial reorthogonalization*. PhD thesis, Dept. Computer Science, University of Aarhus, DK-8000 Aarhus C, Denmark, Oct 1998.

[19] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proc. of the 7th USENIX Security Symp.*, San Antonio, TX, 1998.

[20] W. Lee, S. Stolfo, and K. Mok. Mining in a data-flow environment: Experience in network intrusion detection. In S. Chaudhuri and D. Madigan, editors, *Proc. of the 5th Int'l Conf. on Knowledge Discovery and Data Mining*, pages 114–124, 1999.

[21] E. A. Levy. Smashing the stack for fun and profit. *Phrack magazine*, 7(49), Nov 1996.

[22] 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.

[23] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 2003.

[24] M. Mahoney and P. Chan. Detecting novel attacks by identifying anomalous network packet headers. Technical Report CS-2001-2, Florida Institute of Technology, 2001.

[25] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. on Information and System Security*, 3(4):262–294, 2000.

[26] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proc. 20th NIST-NCSC Nat'l Information Systems Security Conf.*, pages 353–365, 1997.

[27] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report T2R-0Y6, Secure Networks, Calgary, Canada, 1998.

[28] S. Savaresi and D. L. Boley. On the performance of bisecting k-means and PDDP. In *Proc. of the 1st SIAM Conf. on Data Mining*, pages 1–14, 2001.

[29] S. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga. Cluster selection in divisive clustering algorithms. In *Proc. of the 2nd SIAM Int'l Conf. on Data Mining*, pages 299–314, 2002.

[30] G. Serazzi and S. Zanero. Computer virus propagation models. In M. C. Calzarossa and E. Gelenbe, editors, *Tutorials of the 11th IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecom. Systems - MASCOTS 2003*. Springer-Verlag, 2003.

[31] K. Tan and B. Collie. Detection and classification of TCP/IP network services. In *Proc. of the Computer Security Applications Conf.*, pages 99–107, 1997.

[32] 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*, pages 320–324, Aug 2000.

[33] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proc. of the 16th Int'l Conf. on Pattern Recognition*, volume 4, pages 385–388, aug 2002.