

Service-independent payload analysis to improve intrusion detection in network traffic

Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José I. Martín, Javier Muguerza, Jesús M^a Pérez

Dept. of Computer Architecture and Technology
University of the Basque Country
M. Lardizabal, 1, 20018 Donostia, Spain

{inigo.perona, i.gurrutxaga, olatz.arbelaitz, j.martin, j.muguerza, txus.perez}@ehu.es

Abstract

The popularity of computer networks broadens the scope for network attackers and increases the damage these attacks can cause. In this context, Intrusion Detection Systems (IDS) are included as part of any complete security package. This work focuses on nIDSs which work by scanning the network traffic. A service-independent payload processing approach is presented to increase detection rates in non-flood attacks. Three different techniques for payload processing are proposed and they are shown to be able to efficiently detect some of the attack types. Moreover, the proper integration of the knowledge of the different techniques, payload-based and packet header-based, always improves the results. This work leads us to conclude that payload analysis can be used in a general manner, with no service- or port-specific modelling, to detect attacks in network traffic.

Keywords: Intrusion detection systems, unsupervised anomaly detection, payload, AUC.

1 Introduction

The use of computer networks has become very popular lately. This fact broadens the scope for network attackers and increases the damage these attacks can cause. Network attacks compromise the stability of the network or the security of the information stored on computers connected to it. Therefore, it is very important to build systems that are able to detect attacks before they cause damage. Intrusion Detection Systems (IDS) form part of any complete security package and their goal is to detect any action that violates the security policy of a computer system. In this work we will focus on a particular kind of IDS that works by scanning the network traffic and is able to automatically detect intrusions: network Intrusion Detection System (nIDS).

The detection of network attacks by human analysis, where memorization, looking up description libraries or searching sample collections is required, is not effective; it is too time consuming and subjective. As a consequence, automated and robust nIDSs are required in order to successfully confront the problem. In this sense,

data mining techniques have been used mainly to train on labelled data to detect attacks. Analyzing the bibliography, two main approaches can be found in nIDS systems.

The misuse detection approach, which is used in systems such as MADAM/ID (Lee, Stolfo, and Mok 1999) where machine learning techniques are used on labelled data: the classifier learns from a set of labelled connections, where there is normal traffic and attacks, and in subsequent use it recognizes known attacks. These methods have two main problems. On the one hand, it is very difficult to obtain completely labelled network traffic and, on the other hand, these methods are not able to detect new attacks. They need to be revised each time a new kind of intrusion appears and this happens every day. These methods can not solve the “zero-day” problem and as a consequence, the new attacks will always succeed in damaging the system. Nevertheless, the primary objective will be to detect the first occurrence of intrusions and prevent it from damaging any victim.

The second approach, anomaly detection, was originally proposed by Denning 1987 and a survey can be found in Warrender, Forrest and Pearlmutter 1999. This method profiles normal network traffic behaviour and is able to successfully detect attacks when the observed traffic deviates from the modelled behaviour.

When the anomaly detection approach is used, classifiers learn how normal traffic behaves and any anomalous connection is considered to be an attack. As a consequence, if not all the kinds of normal traffic are modelled, high false positive rates can appear in the system. Moreover, they need purely normal data in order to model normal traffic and it is not usual in real systems to have either purely normal data or labelled data. If any attack is left in the hypothetical purely normal data, this attack will be learned as normal traffic and the IDS will never produce an alert related to it.

Taking into account the problems of the two previous approaches, a third one is becoming popular: unsupervised anomaly detection (Portnoy, Eskin and Stolfo 2001). This option works under the assumption that the volume of normal traffic is much greater than the traffic containing attacks, and, furthermore, the intrusions’ behaviour is different from normal data’s behaviour. If these assumptions are true the intrusion detection problem can be confronted in terms of outlier detection. This approach can be used as a stand-alone system, or, even more effectively, it can be combined with a misuse detection or anomaly detection process.

These kinds of systems have two main advantages, they do not need purely normal data and unlabelled data can be used, which is easy to obtain.

Unsupervised anomaly detection methods usually build probabilistic models of the data that will help them to decide whether or not the connections are attacks. In this context, clustering methods can be used as a tool for anomaly detection. The connections will be clustered and instances appearing in small clusters, i.e. anomalies, will be considered intrusions.

The mentioned assumptions make unsupervised anomaly detection methods inadequate for detecting flood attacks. These kinds of attacks usually need to send a large number of packets in a short time, and are used for many kind of “Denial of Service” (DoS) and “Probe” attacks. Since they are based on the emission of many similar packets they will naturally form large groups and, as a consequence, the clustering process will group them in large clusters. Nevertheless, flood attacks are easy to detect and high detection rates can be often achieved by simpler systems that scan network traffic or analyze headers (Noh, Jung, Choi and Lee 2008).

Although most of the flood attacks can be successfully detected by scanning the TCP/IP headers of network packets, this information is not enough to detect most of the non-flood attacks. In these kinds of attacks, mostly “User to Root” (U2R) and “Remote to Local” (R2L), the intruder only has to send very few packets (often, a single one is sufficient) and, as a consequence, it is nearly impossible for systems to use traffic models to detect such anomalies. In this context, the use of the data transported in the packages –payload– becomes crucial to detect intrusions. Notice that R2L and U2R attacks are actually the only ones that allow the intruder to obtain complete control of the attacked system and therefore, they can lead to catastrophic consequences.

The payload of different network connections can be very different. The transferred information usually depends on the kind of service, and, as a consequence its global analysis becomes complicated. This is probably the reason why there are few works where the payload is used to model network traffic and detect the possible intruders. When payload is used with this aim service-specific approaches are developed. For example Krügel, Toth, and Kirda 2002 present a work that focuses on R2L attacks and uses service-specific knowledge to increase the detection rate of intrusions. They have implemented a prototype that can process HTTP and DNS traffic although they only present results for DNS.

Wang, and Stolfo 2004 base their work on profile byte frequency distribution and they compute the standard deviation of the application-level payload flowing to a single host and port during a training phase. The Mahalanobis distance is used during the detection phase and if the distance exceeds a certain threshold the system generates an alarm. This model is also host- and port-specific and also conditioned by the payload length.

These service-specific methods have the disadvantage that they are very context dependent. That is to say, as they are moved to machines offering different services or as new services appear, the system will need to be rebuilt.

Another example of payload processing can be found in the content variables of Kddcup99 (Lee 1999). In this

case, it has been used to obtain some information based on the experts’ experience. This kind of processing is very context dependent and it can only be done for some well known services and protocols. The processing is totally static; it has no learning capability at all. In order for it to be adapted to new situations the experts need to manually analyze the network data and adapt their knowledge to new attacks.

Each kind of method has its advantages and disadvantages: signature-based methods have the advantage of generating few false alarms, whereas anomaly-based systems generally produce a lot of false alarms for unusual but authorized activities, which is not recommendable at all. Anyway, the false negatives (attacks not detected) generated by signature-based methods are far more problematic than the false alarms generated with anomaly detectors.

In order to obtain the best from each kind of intrusion detection system, a combination of some of them is usually the best option. For example, a flood detecting firewall could first filter most flood attacks; a signature-based IDS could then be used to remove the known attacks and unsupervised anomaly detection could finally focus on detecting the unknown attacks.

Our aim in this work is to present an approach where service-independent payload processing can be used to increase detection rates in non-flood attacks. With this aim we have first analyzed different approaches to payload processing in order to see to what extent just the information provided by the payload can be used to detect intrusions. Next, we have evaluated the combination of the payload-based approaches with the information that can be obtained from network packet headers. After this preliminary work it seems that the techniques for payload processing are able to efficiently detect some of the attack types. Furthermore, they can be used to complement techniques based on packet header analysis, since the combinations tried always improve the results.

The paper proceeds to describe, in Section 2, the approximation that will be used in this work for outlier detection. Section 3 is devoted to describing the three global approaches proposed to process payload without any context knowledge. In Section 4 we describe the data used in the experimentation and experimental results are presented in Section 5. Before the conclusions we summarize in section 6 the schema of the proposed system. Finally, Section 7 is devoted to presenting the conclusions and further work.

2 Detecting outliers

As we mentioned in the introduction, unsupervised anomaly detection strategies can be formulated as outlier detection problems. The approximation we have used to detect outliers could be based on any clustering algorithm. The idea is to first perform the clustering over the points in the feature space and assign a score to each of them based on their size. The examples in each cluster will have the same score the cluster has, and this score will be used to determine the degree of anomaly of the example. The points with lower scores will be labelled as anomalous. Although many clustering algorithms could be used, based on the experience of other authors (Eskin, Arnold, Prerau, Portnoy, and Stolfo 2002, Leung, and

Leckie 2005), we have selected the fixed-width clustering algorithm (Eskin, Arnold, Prerau, Portnoy, and Stolfo 2002), also known as the leader algorithm (Spath 1980). The fixed-width algorithm has the advantage that it scales linearly to the number of examples of the database and the number of clusters, but, on the other hand, it has the disadvantage that the quality of the clusters is sensitive to the definition of the radius w and often several runs are needed to select the best w in any particular environment. Moreover, this algorithm does not accurately fit to databases with clusters of different sizes; the largest clusters are usually over-partitioned. Nevertheless, in the unsupervised anomaly detection context we are interested just in the small groups, so this drawback of the algorithm is not a real problem.

3 Payload processing

Network Intrusion Detection Systems usually focus on the analysis of the TCP/IP headers of the packets detected on the net. The format of these headers is well-known and, therefore, this data can be easily processed. This information might not be enough to detect intruders and as a consequence, the analysis of the transferred data, the payload, will need to be performed. The format of the payload data in a packet depends on the application protocol used, so that in this case data processing becomes a difficult task. Moreover, many protocols have fields where any kind of data can be stored. Many previous works have solved this problem by performing the data processing in a specific way for each service. Obviously, this method requires knowledge of the different protocols and has many drawbacks: it only works for a reduced set of connections, the used protocol is not always known, new services are not automatically treated...

Payload data can generally be seen as a sequence of bytes, so we think that it could be processed, regardless of the service used, by using sequence comparison techniques. The aim of this work is to use this payload information as a tool to help detecting attacks in nIDSs. In this context, it is important for the selected payload processing method to be efficient in detecting attacks and to achieve low false positive rates, but, it also requires having some other characteristics such as:

1. To be automatic. That is, not requiring human intervention.
2. To be general. That is to say, service-independent, and, as a consequence, usable in different environments and adaptable to changing situations.
3. To be computationally efficient so that it can operate in real time, in environments with large bandwidth.

It is not easy to build a system with all the required skills; it seems, on the one hand, that more complex or computationally expensive systems would better model the payload, and, on the other hand, that service- and models are easier to build. As we mentioned in the introduction, some work has already been done in this context, as for example Krügel, Toth, and Kirda 2002, Wang, and Stolfo 2004, Leung, and Leckie 2005.

The aim of this work is to build a system that achieves all the required characteristics, as far as possible, and to demonstrate that payload can be used to improve the behaviour of IDSs in a general and computationally effective way. In this context, several payload processing strategies have been tried:

1. The first and probably most intuitive option seems to directly use the payload as a byte sequence and use sequence comparison methods to model different payloads. It can be easily concluded that distances such as the Edit Distance (Gusfield 1997), where the distance between two strings of characters is the minimum number of edit operations on individual characters needed to transform one string to another, are computationally too expensive. As a consequence, this approach will only be useful if a more efficient method for comparing sequences can be used. In this context we have used the Normalized Compression Distance (NCD) proposed in Li, Chen, Li, Ma, and Vitanyi 2004, Wehner 2005. This distance is based on the Kolmogorov complexity. Since in practice Kolmogorov complexity cannot be computed directly, it is approximated with real compression algorithms. In this work we used the standard compression algorithm GZip. $C(x)$ is defined as the length of the string obtained by compressing x , and $C(xy)$ as the length of the string obtained by compressing the concatenation of x and y . The Normalized Compression Distance is defined as follows:

$$NCD(x,y) = \{ C(xy) - \min(C(x), C(y)) \} / \max(C(x), C(y)).$$
2. The well known n-gram analysis is another option that could be used to model payload. To make it computationally efficient a 1-gram model, where the frequency of each one of the 256 values is computed, could be used. This method has been used by some other authors (Wand, and Stolfo 2004). Even the 1-gram option can have some limitations on the size of the database used, since 256 variables \times number of examples can be too large when working with large databases. On the other hand, it is improbable that all 256 variables would be significant in each of the payloads and the treatment of all of them could even be counterproductive.
3. Based on the byte-frequency idea, we think that the sequence of the most frequent bytes could in some sense be more significant for representing the payload. Similar connections will probably have similar payload patterns and as a consequence we could expect also the most frequent bytes to be similar. That is why the 1-gram representation of the payload has been calculated and the 256 possible values ordered from the most frequent to the least frequent. The information referred to the N most used bytes has been used (we have tried 3 different values for N : 15, 30 and 50). The payload byte-ordered

vectors PBO_1 and PBO_2 could be two examples of the representation of two different payloads (P_1 and P_2):

PBO_1	8A	F1	05	AE	87		91
	1	2	3	4	5	...	N
PBO_2	F1	8A	05	AE	90		8C
	1	2	3	4	5	...	N

In this way, a new representation of the payload is achieved which can be used in different ways:

1. Each byte in the new vector could be treated as an independent variable and the Euclidean Distance used to compare two payloads. The distance between two byte values has been set to 1 if they are different and 0 otherwise. We will call this option Freq1.
2. The representation can be considered a sequence where the position in which each character appears influences the distance. In this context, a distance has been defined to compare the representation of two payloads. If two payloads are compared and their corresponding payload character-ordered vectors (PBO_1 and PBO_2) are obtained, the distance could be computed with the expression:

$$Dist(PBO_1, PBO_2) = 1 - \frac{\sum_{i=1}^N Sim(i)}{N^2}$$

where

$$Sim(i) = \begin{cases} 0 & \text{if } PBO_1(i) \neq PBO_2(j), \forall j, 1 \leq j \leq N, j \neq i \\ N - |i - j| & \text{if } \exists j | PBO_1(i) = PBO_2(j), 1 \leq j \leq N, j \neq i \end{cases}$$

We will call this option Freq2.

4 Data generation

As we mentioned in the introduction, it is not easy to obtain labelled data for network traffic, and neither a database with purely normal data. As a consequence, it is also difficult to evaluate intrusion detection systems and compare results. Even if unsupervised anomaly detection techniques do not require labelled data to work, this kind of data is required so that the system can be evaluated. In order to generate comparable results, we have decided to use some standard data such as Kddcup99 from the UCI repository (KDD99-Cup 1999). This database was built by processing the DARPA98 dataset (DARPA 1998), which was generated by the Information System Technology Group (IST) of the Lincoln Laboratory of the MIT with the collaboration of DARPA and ARFL. They built a network to simulate a real situation of network traffic containing normal traffic and attacks. Tcpcdump (Jacobson, Leres, and McCanne 1989) was used to sniff the network and store all the packets belonging to network traffic in a tcpcdump file. Based on this information the UCI format Kddcup99 database was generated by identifying connections and aggregating information belonging to them. Based on the information in DARPA98, three kinds of features were generated for

each connection: intrinsic variables (those obtained by examining the packets' TCP/IP structure such as protocol, length, urgent bit...); content variables which were obtained by examining the payload of some particular services, such as number of failed logins, number of operations generated as root, number of file creations; and, finally, traffic variables which take into account header information of preceding connections contained in a window of some specific size.

The advantage of the KddCup99 database is that it processes the huge amount of information in the DARPA98 dataset and it stores it in a format suitable for most machine learning algorithms. The problem is that the original payload information is not stored, and just some manually defined attributes (content variables) keep part of the payload-based information. This manual solution is obviously not a general solution. Based on Lee 1999 and using Bro (Paxson 1998), we have reprocessed the DARPA98 database to embellish Kddcup99 database with information from the original DARPA98. In this new database, each connection will have the intrinsic and traffic variables found in the original database added to all the payload data corresponding to it. The content variables have not been computed and added to the new database, because the aim of this work is to replace the information they provide by automatic payload processing.

Attacks	Quantity	percentage
anomaly	9	0.23
dict	879	22.33
dict_simple	1	0.03
eject	11	0.28
eject-fail	1	0.03
ffb	10	0.25
ffb_clear	1	0.03
format	6	0.15
format_clear	1	0.03
format-fail	1	0.03
ftp-write	8	0.20
guest	50	1.27
imap	7	0.18
land	35	0.89
load_clear	1	0.03
loadmodule	8	0.20
multihop	9	0.23
perl_clear	1	0.03
perlmagic	4	0.10
phf	5	0.13
rootkit	29	0.74
spy	2	0.05
syslog	4	0.10
teardrop	1085	27.56
warez	1	0.03
warezclient	1749	44.42
warezmaster	19	0.48
Total attacks	3937	

Table 1: Names and quantities of attacks that appear in the database used for experimentation

Due to the huge size of the original Kddcup99 database (about 5,000,000 connections), most of the experiments found in literature have been performed using a sample of the original dataset. This sample contains about the 10% of the connections found in the complete dataset. Similarly, we have extracted a stratified sample of about 10% of the size of the original one. Since our goal is to find the non-flood attacks, and the DARPA98 is overloaded with flood attacks, we have filtered all the flood attacks in the dataset. Thus, we have been working with a database of 178,810 examples, where 3,937 examples belong to intrusions of 27 different kinds. The information about the kind of attacks appearing in the database used for experimentation and their frequency is shown in Table 1.

5 Experimental results

5.1 Evaluation of different representations

Our objective is to use the payload in a general way for unsupervised anomaly detection so that intrusions are differentiated from normal traffic. The idea is to combine the information the payload can provide with the TCP/IP header information (intrinsic and traffic variables in Kddcup99). As we mentioned in Section 3, we have tried several options for processing the information provided by the payload.

The first step has been to analyze how each one of the representations works separately for anomaly detection with no more information than just payload and to compare the results obtained with those obtained with TCP/IP header information i.e. the equivalent of the intrinsic and traffic variables of KddCup99. Each of the representations has been used to build a model of the system using a fixed-width algorithm with an adequate value for w .

The experimentation has been done with the whole database, that is, normal traffic data plus data from 27 different kinds of attacks. But, in order to present the results obtained in our experimentation we have chosen to examine the detection rates of each type of attack separately instead of examining a global result for all of them. We have noticed that in many previous works where Kddcup99 was used for experimentation a single global value was used to measure the accuracy of a particular detection system. Due to the high number of connections belonging to certain attacks (mostly Smurf and Neptune) the results are heavily biased to the detection level the system obtained for those particular attack types. Although we have removed all the flood attacks from our dataset, differences in the number of attacks of each type exist (see Table 1). Our results will be always analyzed for each attack type, and, therefore, all attack types will have the same weight when presenting the final results. Otherwise the overall results would practically ignore the attack types other than Warezclient, Teardrop and Dict. The evaluation has been done by analyzing the ROC curves and the Areas Under ROC Curves, or AUC values, obtained (Fawcett 2004). To compute the ROC of just a single attack type, the examples belonging to other attack types have been ignored.

In general terms, the first conclusion that can be drawn from this processing is that although no context knowledge is used and simple processing is performed, the three options for modelling payload (NCD, Freq1, Freq2) are, to a certain extent, able to differentiate between normal traffic and intrusions; the average AUC values are specifically 0.77 for NCD, 0.74 for Freq1 and 0.72 for Freq2 (see Table 3 for AUC values in each attack). Furthermore, the model built using header information, IT, achieves an AUC value of 0.80, which is not very different from the previous ones. In order to take the conclusions a bit further we have made a pair wise comparison of the 4 different techniques. In each comparison we have compared the AUC values obtained for each attack type and counted the number of times each technique obtains greater values than the other. The cells in Table 2 show the number of attacks (out of 27) the technique of the corresponding row detects better than the technique of the corresponding column. The last column shows the number of times the corresponding technique behaves better than any of the rest.

	IT	NCD	Freq1	Freq2	Total
IT	0	10	14	16	40
NCD	17	0	17	13	47
Freq1	13	10	0	16	39
Freq2	11	14	11	0	36

Table 2: Pair wise comparison of the different techniques. Summary of the number of times each technique behaves better than the other

In order to interpret the results it is important to notice that if a certain technique was always better than another one, the value in the cell corresponding to their comparison would be 27 (values of around 13-14 would mean similar behaviour), and if that happened for all three comparisons that can be made for each classifier, the total value (column Total) would be 81. Values in Table 2 are, in general, not very far from 13. This means that the number of attacks each classifier is able to detect better is similar. However, if we observe the general behaviour of different options it seems that one of them, NCD, behaves better than the rest. This means that, we would be able to detect more attacks with general payload processing, than by analyzing header information (IT).

5.2 Combining results

If the ROC curves and AUC values obtained for each of the attacks are analyzed, it can be observed, that, even if similar detection rates are achieved on average, each one of the classifiers built specializes better in detecting some kinds of attacks. As an example, we have observed that Teardrop, Warez, Format-clear and Warezclient attacks are better detected by IT, NCD, Freq1 and Freq2 approaches respectively (see Table 3).

This leads us to think that a good option could be to combine the knowledge acquired in all of them; but, how could be this combination be done? The output of each of the classifiers built can be seen as a list of all of the classified patterns with their corresponding scores.

Attacks	IT	NCD	Freq1	Freq2	IT-NCD-Freq1	IT-NCD-Freq2	IT-Freq1-Freq2	IT-NCD-Freq1-Freq2
anomaly	0.76	0.88	0.35	0.74	0.72	0.9	0.68	0.77
dict	0.76	0.82	0.64	0.83	0.83	0.93	0.84	0.89
dict_simple	0.65	0.81	0.69	0.83	0.81	0.91	0.81	0.88
eject	0.76	0.82	0.8	0.80	0.9	0.9	0.88	0.92
eject-fail	0.99	0.48	0.99	0.58	0.94	0.8	0.98	0.9
ffb	0.8	0.88	0.72	0.93	0.9	0.97	0.92	0.94
ffb_clear	0.65	0.81	0.71	0.67	0.82	0.84	0.73	0.84
format	0.79	0.93	0.81	0.95	0.95	0.98	0.96	0.98
format_clear	0.52	0.81	0.88	0.83	0.84	0.85	0.84	0.9
format-fail	0.98	0.81	0.8	0.67	0.98	0.95	0.93	0.96
ftp-write	0.88	0.88	0.76	0.56	0.94	0.88	0.82	0.89
guest	0.77	0.85	0.81	0.83	0.92	0.94	0.92	0.95
imap	0.9	0.7	0.97	0.68	0.94	0.85	0.96	0.92
land	0.92	0.48	0.99	0.58	0.9	0.76	0.95	0.87
load_clear	0.65	0.81	0.12	0.14	0.54	0.56	0.19	0.43
loadmodule	0.7	0.71	0.69	0.68	0.77	0.79	0.77	0.8
multihop	0.72	0.78	0.63	0.71	0.76	0.8	0.73	0.77
perl_clear	0.95	0.81	0.52	0.87	0.87	0.98	0.91	0.93
perlmagic	0.66	0.83	0.86	0.86	0.88	0.91	0.9	0.93
phf	0.9	0.71	0.99	0.72	0.96	0.89	0.97	0.95
rootkit	0.88	0.77	0.86	0.77	0.93	0.9	0.94	0.93
spy	0.71	0.81	0.66	0.52	0.8	0.77	0.68	0.77
syslog	0.82	0.48	0.97	0.58	0.87	0.7	0.92	0.84
teardrop	0.96	0.48	0.76	0.58	0.83	0.78	0.88	0.81
warez	0.82	1.00	0.12	0.85	0.68	0.98	0.64	0.82
warezclient	0.81	0.86	0.86	0.86	0.95	0.96	0.96	0.97
warezmaster	0.94	0.87	0.96	0.88	0.98	0.98	0.98	0.99
Average	0.80	0.77	0.74	0.72	0.86	0.87	0.84	0.87

Table 3: AUC values achieved for all the attack types in different classifiers; single ones and combined ones

As we mentioned in section 2, the score assigned to each of the patterns is used to determine whether the pattern is anomalous or not. As a consequence, the most direct way of combining the different techniques will be to combine the different scores obtained for each connection. Since the distribution of the score values assigned by each technique depends on the number and size of the clusters, it varies from one model to another. This fact means that the direct combination of the score values is probably not adequate (this option has been tried and suspicions have been confirmed). As a consequence, some normalization is required to put the scores of the different classifiers in the same situation. Two options for normalizing the scores in each one of the examined techniques have been tried in this work:

1. Linear 0-1 normalization. Normalize all scores linearly, so that the minimum value is 0 and the maximum is 1.
2. Rank normalization. In this normalization all the connections are sorted by their score (connections with equal score are sorted arbitrarily). The rank for each connection will be

its new score. If more than one connection has the same score, the average rank value of the group of connections with equal score is computed and the result assigned as the new score for all the connections in the group.

The second option is more appropriate (as we have observed in the results), since it is more independent of the specific scores obtained by each of the classifiers and it depends on the position each connection has in the ranking.

Independently of the values used to compare the scores, how should we combine the different values? We could probably think of many different combination strategies but for this work we have tried three of them:

1. Select the minimum score for each connection.
2. Select the maximum score for each connection.
3. Average the score of the combined techniques for each connection.

The three proposed combining methodologies achieve reasonable results, but in general, better results have been obtained by averaging the scores.

The last point to decide at this stage is to select which classifiers' results to combine. In the results presented in the previous section we worked with 4 classifiers and we could combine all of them or select pairs or trios and combine them. We tried all the possible combinations between the four techniques evaluated in Section 5.1. The effect of the combination is in most cases positive but it becomes more positive when more than two techniques are combined.

Table 3 shows AUC values achieved for every attack type with each one of the techniques used: the four options used independently and all the possible combinations of three and four techniques where IT (connection header information) is always maintained. IT has not been ignored in any of the combinations because, as we mentioned in the introduction, our aim is to provide our system with both header and payload information. The best AUC for each of the attacks (taking into account all decimal values) is marked in bold.

IT	72
NCD	72
Freq1	61
Freq2	55
IT-NCD-Freq1	114
IT-NCD-Freq2	130
IT-Freq1-Freq2	109
IT-NCD-Freq1-Freq2	143

Table 4: Pair wise comparison of the 8 different techniques. Total number of times each technique behaves better than the other

Average AUC values show that the general behaviour of any of the combinations improves the behaviour of the classifiers obtained with each independent option. We could consider as the best option the combination of the 4 techniques because, even if the average AUC value obtained is similar to the one obtained with the IT-NCD-Freq2 option, it achieves the best AUC for more different kinds of attacks (9 out of 27).

The same kind of pair wise comparison presented in Table 2 has been performed for all the evaluated techniques. The global results —number of times each classifier behaves better than the other out of 189— for the eight techniques evaluated in Table 3, are presented in Table 4 and they confirm the conclusions drawn from Table 3. Results for combinations of two techniques are in general between the results of single techniques and results of the combination of three or more techniques.

6 Schema of Intrusion detection process

For clarity, in this section we summarize the steps the payload based intrusion detection tool we propose needs. Figure 1 shows a schema of the process.

Once network data is collected we will divide it in two main parts: the connections' headers information on the one hand, and the transferred information or payload on the other one. The TCP/IP headers' information will be processed to obtain a tabular representation with intrinsic variables and traffic variables. The payload part will be processed to obtain information about the frequency of the transferred bytes and represent it also in a tabular manner.

Next step is to apply fixed width clustering algorithm with the distances defined in Section 3 to the headers' information, frequency based representation of payload and payload without processing. Four different partitions will be obtained at this point, IT, Freq1, Freq2 and NCD, with the corresponding scores for the connections.

Finally the scores will be combined to obtain the final score for each connection. These scores will be the ones used to determine the degree of anomaly of the connection.

7 Conclusions and further work

In this work we have proposed three different techniques for payload processing in order to use it to detect attacks in network traffic. The three options proposed have been shown to be able to efficiently detect some of the attack types. Although most of the previous works were based on packet header analysis, this work has shown that general payload analysis can also be effective.

Since the different techniques have shown the ability to detect different kinds of attacks, we have decided to combine them. The results obtained have shown that it is possible to integrate the knowledge of the payload-based techniques and the packet-header-based technique and improve the original results. The combination of all the options tried is in particular the one obtaining the best results.

Based on the results presented we can state that in the data set we have used for experimentation, payload analysis can be used in a general manner, with no service- or port-specific modelling, to detect attacks in network traffic. Obviously, this technology needs still to be tested in real environments.

This is a preliminary work where just some of the possible payload information processing options have been tried. Other techniques could be tried in the future. The way in which classifiers can be combined is another area where a deeper analysis can be carried out and more sophisticated approaches tried. The possibility of using other clustering algorithms and the optimization of their parameters is also an area where more work can be done.

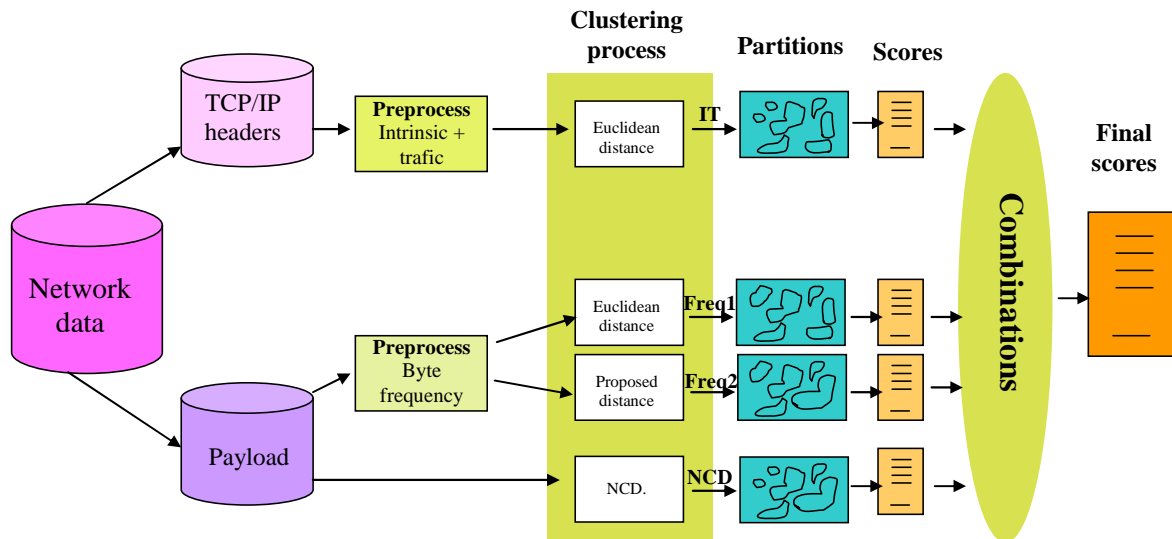


Figure 1: Schema of the proposed intrusion detection process

8 Acknowledgements

This work was partly funded by the Diputación Foral de Gipuzkoa and the E.U.

9 References

- Lee W., Stolfo S.J., Mok K. (1999): Data mining in work flow environments. Experiences in intrusion detection. *Proc. of the Conference on Knowledge Discovery and Data Mining*.
- Denning D.E. (1987): An intrusion detection model. *IEEE Transactions on Software Engineering* **13**:222-232.
- Warrender C., Forrest S., Pearlmutter B. (1999): Detecting intrusions using system calls: alternative data models. *Proc. IEEE Symposium on Security and Privacy*, 133-145.
- Portnoy L., Eskin E., Stolfo S (2001): Intrusion detection with unlabeled data using clustering. *Proc. ACM Workshop on Data Mining Applied to Security*.
- Noh S., Jung G., Choi K., Lee C. (2008): Compiling network traffic into rules using soft computing methods for the detection of flooding attacks, *Applied Soft Computing* **8**(3):1200-1210.
- Krügel C., Toth T., Kirda E. (2002): Service specific anomaly detection for network intrusion detection. *Proc. ACM Symposium on Applied Computing*, Madrid, Spain, 201-208, ACM Press.
- Wang K., Stolfo S. (2004): Anomalous payload-based network intrusion detection. *Proc. International Symposium on Recent Advances in Intrusion Detection*, LNCS, 203-222.
- Lee W. (1999): A data mining framework for constructing features and models for intrusion detection systems. Ph.D. thesis. Columbia University.
- Eskin E., Arnold A., Prerau M., Portnoy L., Stolfo S. (2002): A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Data Mining for Security Applications*.
- Leung K., Leckie C. (2005): Unsupervised anomaly detection in network intrusion detection using clusters. *Proc. Australian Computer Science Conference*.
- Spath H. (1980): *Cluster analysis algorithms*. Ellis Horwood, Chichester, UK.
- Gusfield D. (1997): *Algorithms on strings, trees, and sequences*. Cambridge University Press.
- Li M., Chen X., Li X., Ma B., Vitanyi P.M.B. (2004): The similarity metric. *IEEE Transactions on Information Theory* **50**:3250-3264.
- Wehner S. (2005): Analyzing worms and network traffic using compression. Technical Report. National research institute for mathematics and computer science in the Netherlands.
- KDD99-Cup (1999): The third international knowledge discovery and data mining tools competition dataset <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Accessed 29 Sep 2008.
- DARPA (1998): MIT Lincoln Laboratory - DARPA Intrusion Detection Evaluation <http://www.ll.mit.edu/IST/ideval/index.html>. Accessed 29 Sep 2008.
- Jacobson V., Leres C., McCanne S. (1989): Tcpcdump. Available via anonymous ftp to <ftp://ee.lbl.gov>. Accessed 29 Sep 2008.
- Paxson V. (1998). Bro: a system for detecting network intruders in real-time. *Computer Networks* **31**:23-24.
- Fawcett T. (2004): ROC graphs: notes and practical considerations for researchers. Technical Report HPL-2003-4. HP Laboratories, Palo Alto, CA, USA.