Histogram Based Payload Processing for Unsupervised Anomaly Detection Systems in Network Intrusion

Iñigo Perona, Iñaki Albisua, Olatz Arbelaitz, Ibai Gurrutxaga, José I. Martín, Javier Muguerza, and Jesús M. Pérez

Dept. of Computer architecture and Technology
University of the Basque Country
M. Lardizabal, 1, 20018 Donostia, Spain
{inigo.perona,inaki.albisua,olatz.arbelaitz,i.gurrutxaga,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, any complete security package includes a network Intrusion Detection System (nIDS). This work focuses on nIDSs which work by scanning the network traffic. We present a service-independent payload processing approach, based on histogram representation, to increase detection rates in non-flood attacks. We implemented three different options combining histogram representation and fixed width clustering algorithm for anomaly detection, and compared them to a sysstem based on packets' header information, another system based on ad hoc payload processing and our previous general payload processing proposal. The new options outperformed the previous ones; they detected efficiently most of the attack types. Moreover, the proper integration of the knowledge of the different techniques, payload-based and packet header-based, always improved 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.

Key words: Intrusion detection systems, unsupervised anomaly detection, payload, histogram, AUC

1 Introduction

There has been a huge increase in the use of computer networks. This fact broadens the scope for network attackers and increases the damage these attacks can cause. Network attacks affect the security of the information stored on computers connected to the network and its stability. Therefore, it is very important to build systems that are able to detect attacks before they cause damage. Any complete security package includes a network Intrusion Detection System (nIDS).

The detection of network attacks can be done by human analysis or automatically. The detection by human analysis requires memorisation, looking up

description libraries or searching sample collections and it is not effective; it is too time consuming and subjective. As a consequence, in order to successfully confront the problem, the security systems require automated and robust nIDSs. In this sense, a very popular option is the use of data mining techniques, mainly trained on labelled data, to detect attacks. We can find in bibliography three main approaches for nIDSs. The first two are misuse detection approach [11] and anomaly detection approach [26]. Due to the problems the previous approaches have, a third one appeared: unsupervised anomaly detection [18]. Usually the best option is a combination of some intrusion detection systems. 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.

The characteristics of the attacks change depending on the kind of attack and as a consequence, the suitability of a tool to detect them will also change. Most of the flood attacks can be successfully detected by scanning the TCP/IP headers of network packets but this information is not enough to detect most of the non-flood attacks. It is nearly impossible for systems to use traffic models to detect User to Root (U2R) or Remote to Local (R2L) attacks because the intruder only has to send very few packets (often, a single one is enough). R2L and U2R attacks can lead to catastrophic consequences because they are actually the only ones that allow the intruder to obtain complete control of the attacked system. In this context some authors propose the use of another source of information: the transferred information or payload. The features of the payload vary depending on the kind of network connection and service. As a consequence, most payload based IDSs we can find in bibliography are service-specific [10][25][12]. These service-specific methods 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. In this context it would be important to be able to build a system that is able to work in any environment independently of the kind of services or machines.

The aim of our work is to contribute to build efficient and context independent nIDSs. In a previous study [17] we proved that information obtained from general payload processing is useful to detect non flood network attacks in an unsupervised anomaly detection context. But we wanted to go further and the purpose of this work was to answer the following questions. Can general payload processing be more efficient than some specific payload processing for intrusion detection in an unsupervised anomaly detection context? and can this be done in a computationally efficient way?

To answer to our research questions we processed the payloads as byte histograms and used histogram comparison methods [20] to compare different payloads. We used clustering as anomaly detection technique and compared the results achieved with our system, general payload processing, to the ones achieved with specific payload processing. Based on previous experience, we combined results obtained from payload analysis with the results obtained from packet header analysis. The results showed that representing the payload as byte his-

tograms and using histogram comparison methods as distances for clustering builds efficient anomaly detection systems for network attack detection. Furthermore, they can be used to complement techniques based on packet header analysis, since the combinations tried always improved the results.

The paper proceeds to describe in Section 2 the main automatic approaches used to build IDSs. In Section 3 we describe the approximation we used in this work for outlier detection. Section 4 is devoted to describing the histogram signature based approach we used to process payload without any context knowledge. In Section 5 we present the schema of the proposed system. The paper continues in Section 6 where we describe the data used in the experimentation and we present experimental results in Section 7. Before the conclusions Section 8 is devoted to a short discussion. Finally, we summarise in Section 9 the conclusions and further work.

2 Main approaches for IDS

We found in bibliography three main data mining based approaches for intrusion detection: misuse detection, anomaly detection and unsupervised anomaly detection. In the misuse detection approach, which is used in systems such as MADAM/ID [11] the authors use machine learning techniques on labelled data: the classifier learns from a set of labelled connections, where there is normal traffic and attacks, and in subsequent uses it recognises 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, they can not solve the zero-day problem and as a consequence, the new attacks will always succeed in damaging the system. They need to be revised each time a new kind of intrusion appears and this happens every day. Nevertheless, the primary objective will be to detect the first occurrence of intrusions and prevent it from damaging any victim.

Warrender, Forrest and Pearlmutter wrote a survey [26] of IDSs based on anomaly detection approaches. This method profiles normal network traffic behaviour and successfully detects attacks when the observed traffic deviates from the modelled behaviour. In anomaly detection approach, classifiers learn how normal traffic behaves and any anomalous connection is considered to be an attack. As a consequence, if the engineers do not model all the kinds of normal traffic, the systems will have high false positive rates. Moreover, in real environments it is not usual to have purely normal data and these approaches need it in order to model just normal traffic. If any attack is left in the hypothetical purely normal data, this attack will be learnt as normal traffic and the IDS will never produce an alert related to it.

Due to the problems the previous approaches have, many researchers are working on a third one: unsupervised anomaly detection [18]. It does not need purely normal data and it uses unlabelled data, which is easy to obtain. 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. Under these assumptions the

intrusion detection problem can be confronted using outlier detection techniques. 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.

Unsupervised anomaly detection methods are inadequate for detecting flood attacks because these kinds of attacks usually need to send a large number of packets in a short time and as a consequence they will naturally form large groups that will not be detected as anomalies. Nevertheless, flood attacks are easy to detect and some authors achieve high detection rates by simpler systems that scan network traffic or analyse headers [15]. Although it is long since the fist anomaly detection approaches appeared, it is still a successful approach being used in many systems. An example of the use of this methodology is the number of papers mentioning it in the last conference in Recent Advances in Intrusion Detection RAID 2008. Ashfaq et al [1] for example presented a comparative evaluation of 8 lately developed anomaly detectors under portscan attacks from the accuracy, scalability, complexity and detection delay point of view. The authors built two independently collected datasets for the evaluation, both of them including packet header information since all the evaluated systems are based on this information. On the other hand, Dagorn presents in [2] an anomaly-based intrusion detection system for web applications and Rehák eta al [19] present a way to improve error rate in anomaly detection by collective trust modelling.

3 Detecting outliers

As we mentioned in previous section, unsupervised anomaly detection strategies can be formulated as outlier detection problems [6]; they usually build probabilistic models of the data that will help them to decide whether or not the connections are attacks. We concretely used clustering as a tool for anomaly detection. We first performed the clustering over the points in the feature space, the connections, and assigned a score to each cluster based on its size. Then we scored the examples in each cluster based on this score, and we used this score to determine the degree of anomality of the example. We labelled the points with lower scores as anomalous. Although many clustering algorithms could be used, based on the experience of other authors [4][13], we selected the fixed-width clustering algorithm [4], also known as the leader algorithm [21]. The fixed-width scales linearly to the number of examples of the database and the number of clusters. This algorithm does not accurately fit to databases with clusters of different sizes; it over partitions the largest clusters. Nevertheless, in the unsupervised anomaly detection context we are interested just in the small clusters, so this drawback of the algorithm is not a real problem.

4 Payload processing

TCP/IP headers of the packets detected on the network traffic can be easily processed because the format of these headers is well-known. On the contrary, payload processing is a difficult task because its format in a packet depends on

the application and used protocol. Moreover, many protocols have fields where any kind of data can be stored. Some authors solved this problem by performing the data processing in a specific way for each service [10][25][11]. This option has many drawbacks: it works for a reduced set of connections, the used protocol is not always known and new services are not automatically treated. The selected payload processing method needs to be helpful to detect attacks, but, it also requires having some other characteristics such as:

1. Not requiring human intervention. That is, to be automatic.
2. To be service-independent, and, as a consequence, usable in different environments and adaptable to changing situations. That is, to be general.
3. To be computationally efficient.

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. On the contrary, payload data can generally be seen as a sequence of bytes, so in a previous work [17] we already processed it regardless of the kind of service or port, based on byte frequencies and sequence comparison techniques. In this work we processed the payload in a very simple and efficient way: as byte histograms or 1-grams. We represented the ASCII characters (0-255 bins) in the x-axis and their frequencies, normalised with the payloads' length, in the y-axis.

Histogram comparison methods

Histogram comparison is an important field in pattern classification and data clustering. As a consequence there are many approaches to calculate distances or similarities between histograms. Some approaches propose distances for ordered histograms. Strelkov presented in [22] for example a distance based on the closeness of positions and shapes of peaks in the compared histograms. This peak matching measure mainly moves one histogram relatively to another in its inner compute. This kind of distance requires ordered histograms, i.e., the neighbour bins on a histogram need to contain related information. In our case, the histograms are nominal. When processing payload the bins correspond to ASCII characters which are independent from each other. We based our research mainly in the work of Serratosa and Sanfeliu [20]. They propose the use of signatures, a loss-less representation of histograms, to calculate distances between histograms in an efficient way. The signature is a vector that contains the non-zero bins of the corresponding histogram and x-axis indexes for the saved bins. Thus, the signature does not lose information. For each payload (P_l) corresponding to each of the network connetcions we built an histogram $H(P_l) = [H_1(P_l), ..., H_T(P_l)]$ where T is the amount of different discrete values, 256 in our case, and, $H_i(P_l), 1 <= i <= T$ are bins (frequency of the byte i in the payload P_l). Let $S = [S_1(P_l), ..., S_z(P_l)]$ be the signature of the set P_l. Each $S_k(P_l), 1 <= k <= z <= T$, is composed of a pair of terms, $S_k(P_l) = \{w_k, m_k\}$. The first term,w_k, shows the relation between the signature $S(P_l)$ and the histogram $H(P_l)$. Thus, if $w_k = i$ then the second term,

$m_k = H_i(P_l)$ where $m_k > 0$. To compute the distance between two signatures they need to have the same length. The authors propose the use of extended signatures: a signature with the minimum number of empty bins added so that, given a pair of signatures to be compared, the number of bins is the same in both of them. Moreover, each bin in both signatures represents the same bin in the histograms. Once we got the extended signatures we computed the distance between two histograms using Euclidean distance, or using the nominal, Manhattan, or the ordinal, Landmover, distances proposed in [20].

1. The Manhattan distance between two histograms is the number of elements that do not overlap.
$$D_{nom}(S(A'), S(B')) = \sum_{i=1}^{z'} \left| m_i^{A'} - m_i^{B'} \right|$$
2. The Landmover distance between histograms is the minimum number of unit-bin movements needed to transform one histogram to the other. This distance takes into account the sky-line of histograms and can be seen as a way of comparing smoothed histograms [23][8].
$$D_{ord}(S(A'), S(B')) = \sum_{i=1}^{z'-1} \left(w_{i+1}^{A'} - w_i^{A'}\right) \left| \sum_{j=1}^{i} (m_j^{A'} - m_j^{B'}) \right|$$

5 Schema of Intrusion detection process

For clarity, in this section we summarise the steps of the payload based intrusion detection tool we propose. Figure 5 shows a schema of the process. Once network data is collected we divide it in two main parts: the connections headers' information on the one hand, and the transferred information or payload on the other one. We processed the TCP/IP headers information to obtain a tabular representation with intrinsic variables and traffic variables and obtained byte histograms from the payload part. In next step we applied fixed width clustering algorithm to both parts: combined with Euclidean distance for the headers information and combined with the histogram based distances defined in Section 4 for payload. The output of each clustering process was a different partition. Finally, we combined the scores and obtained the final value for each connection. These scores will be the ones used to determine the degree of anomality of the connection.

6 Data generation

It is difficult to obtain labelled data or a database with purely normal data for network traffic. This makes difficult the evaluation and comparison of results of intrusion detection systems; unsupervised anomaly detection techniques do not require labelled data to work, but they need it so that the system can be evaluated. We wanted to generate comparable results and we decided to use some standard data such as Kddcup99 from the UCI repository [9]. Kddcup99 was built from the DARPA98 dataset [3], which was generated by the Information System Technology Group (IST) of the Lincoln Laboratory of the MIT

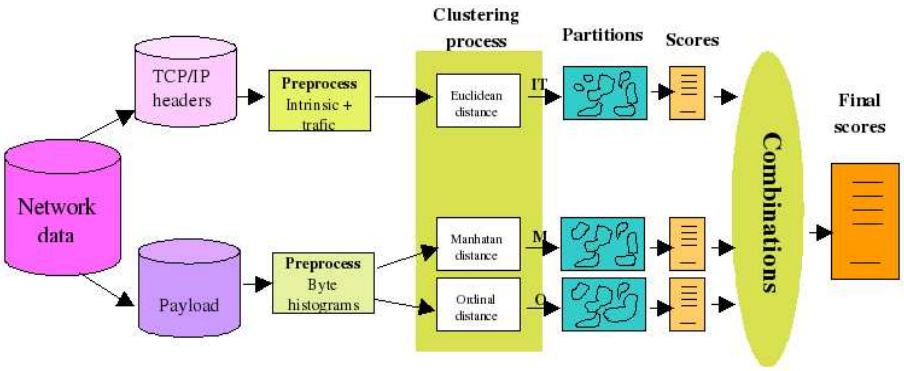

Fig. 1. Schema of the proposed Intrusion Detection System

with the collaboration of DARPA and ARFL. They built a network to simulate a real situation of network traffic containing normal traffic and attacks. They used Tcpdump [7] to sniff the network and stored all the packets belonging to network traffic in a tcpdump file. Lee generated the UCI format Kddcup99 database identifying connections and aggregating information belonging to them. Each connection has three kinds of features: intrinsic variables (those obtained by examining the packets' TCP/IP structure such as protocol, length, urgent bit); traffic variables which take into account header information of preceding connections contained in a window of some specific size; and, finally, content variables obtained by examining the payload of some particular services, such as number of failed logins, number of file creations, etc.. KddCup99 database processes a huge amount of information from DARPA98 dataset and stores it in a format suitable for most machine learning algorithms, but, it does not store the original payload information. The only payload based information it keeps is in the content variables. This is obviously not a general solution.

We reprocessed the DARPA98 database, based on Lee 1999 and using Bro [16], to add information from the original DARPA98 to Kddcup99 database. In this new database, each connection will have the intrinsic and traffic variables of Kddcup99 added to all the payload data corresponding to it. The aim of this work was to replace the information the content variables provide by automatic payload processing.

Due to the huge size of the original Kddcup99 database (about 5,000,000 connections), most authors performed their experiments using a sample of the original dataset. This sample contains about the 10% of the connections. Similarly, we 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 filtered all the flood attacks in the dataset. Thus, we worked with a database of 178,810 examples, where 3,937 examples belong

to intrusions of 27 different kinds. We show in the first two columns in Table 1 the information about the kind of attacks and their frequency.

7 Experimental results

In a previous work [17] we proved that the payload on its own could be used to detect intrusions. In this work we evaluated the performance of the histogram-based payload representation and the three distances described in Section 4. As first approximation we used just the payload information to build classifiers based on the methodology described in Section 3. We built models of the system using fixed-width algorithm for 7 different options:

1. The specific content variables defined in Kddcup99 (C).
2. Three options experimented in our previous work [17].
   - Payload treated as sequence of bytes and compared with NCD distance [14](NCD).
   - Payload represented with the 30 most frequent bytes. We treated each byte as independent variable and used Hamming distance to compare payloads. We called this option MFN (Most Frequent Nominal).
   - Payload represented with the 30 most frequent bytes and considering it a sequence where the position of each character influences the distance. We called this option MFO (Most Frequent Ordinal).
3. Three options for comparing histograms described in Section 4
   - Euclidean distance (HE).
   - Manhattan distance (HM).
   - Ordinal distance (HO).

We experimented with the whole database, that is, normal traffic data plus data from 27 different kinds of attacks. We evaluated the results based on the ROC curves and the Areas Under ROC Curves, or AUC values, obtained [5]. To compute the ROC of just a single attack type, we ignored the examples belonging to other attack types. For each option we present detection rates of each type of attack separately, minimum AUC achieved with each model and weighted average —taking into account the number of attacks of each type— of the achieved AUC. We also generated and included results obtained using just intrinsic and traffic variables (IT) as baseline.

The rows in Table 1 belong to differend attack types whereas the columns belong to different systems. The second column in Table 1 shows the number of examples of each type of attack we find in the database, the third one shows AUC values achieved using packet header information (IT) and, next seven columns show the results achieved with each methodology based only in payload information.

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 six options for modelling payload in a general way (NCD, MFN, MFN, HE, HM, HO) are able to differentiate between normal traffic and intrusions and besides, the three options we proposed in this work, HM, HE and HO, achieved

the highest average AUC values, they did it even better than the model built with data obtained using context specific knowledge for processing payload (C) or the packets' header information (IT). Besides, the row with minimum AUC value for each option shows that mainly two of the options are more interesting: HM and HO. They achieved AUC values of 0.69 or bigger for all the kinds of attacks. The rest of the options have minimum values smaller than 0.5 which means that for some kind of attack they achieved worse results than a random classifier would.

Different techniques showed the ability to detect different kinds of attacks, and based on previous experience, we knew that it is possible to integrate the knowledge of the payload based techniques and the packet header based technique to improve the original results. Thus, we combined by averaging scores [17] the results obtained with Intrinsic and Traffic variables (IT), with the results obtained with Content variables on the one hand (IT+C), and, with results obtained with HM and HO (IT+HM+HO) on the other hand; the best two options of histogram processing. The combinations contributed to increase the overall AUC values in both cases. The combination of IT, HM and HO is the one that achieved the best results with average AUC of 0.955 and minimum AUC value of 0.89.

8 Discussion

The experimentation presented in previous section proves that a general payload processing methodology, histogram representation, is more efficient than the specific payload processing done in Kddcup99 [12] for intrusion detection in an unsupervised anomaly detection context. The proposed payload processsing options were able to detect more attacks in both cases: when used on their own and when combined with the information of intrinsic and traffic variables. And besides, the proposed options are not computationally expensive since they require simple mathematical operations.

The payload of different network connections can be very different. The transferred information usually depends on the kind of service, and, as a consequence 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 [10] presented a work that focuses on R2L attacks and uses service-specific knowledge to increase the detection rate of intrusions. They implemented a prototype that can process HTTP and DNS traffic although they only presented results for DNS. Wang, and Stolfo [25] based their work on profile byte frequency distribution and they computed the standard deviation of the application-level payload flowing to a single host and port during a training phase. They used the Mahalanobis distance during the detection phase and if the distance exceeded a certain threshold the system generated an alarm. This model is also host- and port-specific and conditioned by the payload length. In a different context, Wazumi et al. in [24] also processed the payload as byte histograms for early worm detection. But they did a different work since, instead of concentrating in the reduction of false

Table 1. Attack detection rates for different models

attacks		IT	C	NCD	MFN	MFO	HM	HE	HO	IT+C	IT+HM+HO
anomaly	9	0.76	1	0.88	0.35	0.74	0.98	0.64	1	1	0.96
dict	879	0.76	0.99	0.82	0.64	0.83	0.93	0.92	1	0.95	0.95
dict_simple	1	0.65	1	0.81	0.69	0.83	0.98	0.99	1	1	0.96
eject	11	0.76	0.98	0.82	0.8	0.8	0.99	0.35	0.98	0.98	0.96
eject-fail	1	0.99	0.8	0.48	0.99	0.58	0.95	0.95	0.97	1	0.96
ffb	10	0.8	0.85	0.88	0.72	0.93	0.95	0.65	0.96	0.93	0.95
ffb_clear	1	0.65	1	0.81	0.71	0.67	0.99	0.97	0.96	1	0.95
format	6	0.79	0.75	0.93	0.81	0.95	0.93	0.82	0.96	0.89	0.93
format_clear	1	0.52	1	0.81	0.88	0.83	0.99	0.21	0.95	1	0.92
format-fail	1	0.98	1	0.81	0.8	0.67	0.99	0.75	0.95	1	1
ftp-write	8	0.88	0.73	0.88	0.76	0.56	0.8	0.87	0.9	0.87	0.89
guest	50	0.77	1	0.85	0.81	0.83	0.93	0.92	0.89	0.94	0.96
imap	7	0.9	0.8	0.97	0.97	0.68	0.97	0.86	0.89	0.92	0.94
land	35	0.92	0.8	0.48	0.99	0.58	0.95	0.95	0.88	0.94	0.94
load_clear	1	0.65	1	0.81	0.12	0.14	0.93	0.99	0.87	1	0.92
loadmodule	8	0.7	0.84	0.71	0.69	0.68	0.97	0.77	0.87	0.87	0.89
multihop	9	0.72	0.74	0.78	0.63	0.71	0.93	0.94	0.84	0.83	0.98
perl_clear	1	0.95	1	0.81	0.52	0.87	0.99	0.99	0.81	1	0.94
perlmagic	4	0.66	1	0.83	0.86	0.86	0.99	0.99	0.77	1	0.96
phf	5	0.9	0.5	0.71	0.99	0.72	0.98	0.98	0.77	0.88	0.93
rootkit	29	0.88	0.81	0.77	0.86	0.77	0.94	0.96	0.76	0.87	0.96
spy	2	0.71	0.8	0.81	0.66	0.52	0.99	1	0.76	0.86	0.98
syslog	4	0.82	0.8	0.48	0.97	0.58	1	0.9	0.75	0.85	0.94
teardrop	1085	0.82	0.65	0.48	0.76	0.58	0.89	0.48	0.69	0.85	0.91
warez	1	0.96	0.31	1	0.12	0.85	0.93	0.97	0.69	0.98	0.96
warezclient	1749	0.81	0.68	0.86	0.86	0.86	0.94	0.92	0.69	0.83	0.95
warezmaster	19	0.94	0.75	0.87	0.96	0.88	0.82	0.87	0.69	0.96	0.95
min		0.52	0.31	0.48	0.12	0.14	0.8	0.21	0.69	0.83	0.89
Average		0.845	0.80	0.746	0.631	0.765	0.929	0.918	0.854	0.91	0.955

positives, and as a consequence the AUC in a network, they proposed a payload processing methodology to detect worms in different networks; they only experimented with a worm, Beagle_AV. Another example of payload processing can be found in the content variables of Kddcup99 [12]. In this case, the author obtained some information from the payload 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 analyse the network data and adapt their knowledge to new attacks. We presented in a previous work in nIDSs [17] three different techniques for payload processing. The three options were able to efficiently detect some of the attack types. This work showed that general payload analysis can be effective but the best results were always achieved including NCD sequence comparison method [14] for payload processing. The present work improves it in two senses:

- Any of the histogram based payload processing strategies achieved greater AUC values than NCD, MFN or MFO options. They even outperformed the results achieved with specific payload processing, Content variables obtained adhoc for the Kddcup99 database based on experts' experience.
- The new option is computationally cheaper

9 Conclusions and further work

The experimentation presented in this paper proves that a general payload processing methodology, histogram representation, is more efficient than the specific payload processing done in Kddcup99 for intrusion detection in an unsupervised anomaly detection context. The proposed payload processing options were able to detect more attacks in both cases: when used on their own and when combined with the information of intrinsic and traffic variables. The best option, IT+HM+HO, achieved an average AUC of 0.955 whereas the average AUC achieved with the best option for adhoc processing, IT+C, was 0.91. Furthermore, it was able to detect any kind of attacks because the minimum AUC, taking into account all types of attacks, was 0.89. And besides, the proposed options were not computationally expensive since they require few mathematical operations.

The way in which classifiers can be combined is an area where a deeper analysis can be carried out and more sophisticated approaches tried. The possibility of using other clustering algorithms and the optimisation of their parameters is also an area where more work can be done. In the same way, we could make the system computationally more efficient by minimising the amount of information kept on the signatures. We could keep just the bins higher than concrete threshold value in the signature. But in this case we would lose information so we would need to evaluate the trade-off.

Acknowledgements. This work was funded by the University of the Basque Country, DAMISI project (EHU 08/39), the Diputacin Foral de Gipuzkoa and the E.U.

References

1. Ashfaq B., Robert Ma . J., Mumtaz A., Ali M. Q., Sajjad A., Khayam A.: A Comparative Evaluation of Anomaly detectors under portscan Attacks. Proc. of RAID 2008. (2008)
2. Dagorn N.: WebIds: A Cooperative Bayesian Anomaly Based Intrusion Detection System for Web Applications. Proc. of RAID 2008. (2008)
3. DARPA: MIT Lincoln Laboratory - DARPA Intrusion Detection Evaluation, `http://www.ll.mit.edu/IST/ideval/index.html`. Accessed 29 Sep 2008. (1998)
4. Eskin E., Arnold A., Prerau M., Portnoy L., Stolfo S.: A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabelled data. Data Mining for Security Applications. (2002)
5. Fawcett T.: ROC graphs: notes and practical considerations for researchers. Technical Report HPL-2003-4. HP Laboratories, Palo Alto, CA, USA (2004)

6. Hodge V.J., Austin J.: A Survey of Outlier Detection Methodologies. Artificial Intelligence Review 22, 85–126 (2004)
7. Jacobson V., Leres C., McCanne S.: Tcpdump. Available via anonymous ftp to `ftp.ee.lbl.gov`. Accessed 29 Sep 2008. (1989)
8. Kamarainen J.K., Kyrki V., Ilonen J., Kälviäinen H. : Improving similarity measures of histograms using smoothing projections. Pattern Recognition Letters 24, 2009-2019 (2003)
9. KDD99-Cup: The third international knowledge discovery and data mining tools competition dataset, `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. Accessed 29 Sep 2008. (1999)
10. Krügel C., Toth T., Kirda E.: Service specific anomaly detection for network intrusion detection. Proc. ACM Symposium on Applied Computing. (2002)
11. Lee W., Stolfo S.J., Mok K.: Data mining in work flow environments. Experiences in intrusion detection. Proc. of the Conference on Knowledge Discovery and Data Mining. (1999)
12. Lee W.: A data mining framework for constructing features and models for intrusion detection systems. Ph.D. thesis, Columbia University (1999)
13. Leung K., Leckie C.: Unsupervised anomaly detection in network intrusion detection using clusters. Proc. Australian Computer Science Conference (2005)
14. Li M., Chen X., Li X., Ma B., Vitanyi P.M.B.: The similarity metric. IEEE Transactions on Information Theory 50, 3250–3264 (2004)
15. Noh S., Jung G., Choi K., Lee C.: Compiling network traffic into rules using soft computing methods for the detection of flooding attacks. Applied Soft Computing 8(3), 1200–1210 (2008)
16. Paxson V.: Bro: a system for detecting network intruders in real-time. Computer Networks 31, 23–24 (1998)
17. Perona I., Gurrutxaga I., Arbelaitz O., MartínJ.I., Muguerza J., Pérez J.M.: Service-independent payload analysis to improve intrusion detection in network traffic. Proc. of the 7th Australasian Data Mining Conference (AusDM08). (2008)
18. Portnoy L., Eskin E., Stolfo S.: Intrusion detection with unlabeled data using clustering. Proc. ACM Workshop on Data Mining Applied to Security. (2001)
19. Rehák M., Pechoucek M., Bartos K., Grill M., Celeda P., Krmícek V.: Improving Anomaly Detection Error Rate by Collective Trust Modeling. Proc. of RAID 2008. (2008)
20. Serratosa F., Sanfeliu A.. Signatures versus histograms: Definitions, distances and algorithms. Pattern Recognition 39, 921–934 (2006)
21. Spath H.: Cluster analysis algorithms. Ellis Horwood, Chichester, UK (1980)
22. Strelkov, V.V.: A new similarity measure for histogram comparison and its application in time series analysis. Pattern Recognition Letters 29, 1768–1774 (2008)
23. Sung-Hyuk Cha, Sargur N. Srihari: On measuring the distance between histograms. Pattern Recognition 35, 1355–1370 (2002)
24. Waizumi Y., Tsuji M., Tsunoda H., Ansari N., Nemoto Y.: Distributed Early Worm Detection Based on Payload Histograms. Proc of the IEEE International Conference ICC'07. (2007)
25. Wang K., Stolfo S.: Anomalous payload-based network intrusion detection. Proc. International Symposium on Recent Advances in Intrusion Detection. (2004)
26. Warrender C., Forrest S., Pearlmutter B.: Detecting intrusions using system calls: alternative data models. Proc. of IEEE Symposium on Security and Privacy. (1999)