International Journal of Neural Systems
©World Scientific Publishing Company

AN INTELLIGENT INTRUSION DETECTION SYSTEM FOR HONEYNET DATA

Álvaro Herrero[1]
Department of Civil Engineering, University of Burgos, Burgos (Spain)

Urko Zurutuza[2]
Electronics and Computing Department, Mondragon University, Arrasate-Mondragon (Spain)

Emilio Corchado[3]
Departamento de Informática y Automática, Universidad de Salamanca, Salamanca (Spain)

This study presents a novel intelligent system that provides network managers with a synthetic and intuitive representation of the situation of the monitored network, in order to reduce the widely known high false-positive rate associated to misuse-based Intrusion Detection Systems (IDSs). This proposal relies on the idea that neural projection techniques such as Exploratory Projection Pursuit can adaptively map high-dimensional data into a low-dimensional space, for the user-friendly visualization of data collected by different security tools. The neural system is based on the use of different projection and unsupervised methods for the visual inspection of honeypot data, and may be seen as a complementary network security tool that sheds light on internal data structures through visual inspection. Furthermore, it is intended to understand the performance of Snort (a well-known misuse-based IDS) through the visualization of attack patterns. This may lead to a subsequent improvement of Snort detection rates by updating its signatures. Empirical verification and comparison of the proposed projection methods are performed in a real domain where two different case studies are defined and analyzed. To check the proposed visualization a 1-month dataset and a 5-month data set were captured and visualized for analysis. Experiments proved that whereas a misuse-based IDS may only identify a low percentage of the malicious traffic, a deeper understanding of attack patterns by the proposed system could easily be gained by means of visual inspections.

Keywords: Artificial Neural Networks, Unsupervised Learning, Projection Models, Network & Computer Security, Intrusion Detection, Honeypots.

1 Introduction

A network attack or intrusion will inevitably violate one of the three computer security principles -availability, integrity and confidentiality- by exploiting certain vulnerabilities such as Denial of Service (DoS), Modification and Destruction.[1] One of the most harmful issues of attacks and intrusions, which increases the difficulty of protecting computer systems, is precisely the ever-changing nature of attack technologies and strategies.

For that reason alone, among others, IDSs[2, 3, 4] have become an essential asset in addition to the computer security infrastructure of most organizations. In the context of computer networks, an IDS can roughly be defined as a tool designed to detect suspicious patterns that may be related to a network or system attack. Intrusion Detection (ID) is therefore a field that focuses on the identification of attempted or ongoing attacks on a computer system (Host IDS - HIDS) or network (Network IDS - NIDS).

Visual inspection of traffic patterns is an alternative and crucial aspect in network monitoring.[5] Visualization is a critical issue in the computer network defense environment, which serves to generate a synthetic and intuitive representation of the current situation for the network manager. As a result, several research initiatives have recently applied information visualization to this challenging task.[6, 7, 8, 9] Visualization techniques typically aim to make the available statistics supplied by traffic-monitoring systems more understandable in an interactive way. They therefore focus on traffic data as well as on network topology. Regardless of their specific characteristics, these methods all map high-dimensional feature data into a low-dimensional space for presentation purposes. The baseline of the research

presented in this study is that Artificial Neural Networks (ANNs),[10, 11, 12, 13] in general, and unsupervised connectionist models, in particular, can prove quite adequate for the purpose of network data visualization through dimensionality reduction.[14, 15, 16] As a result, unsupervised projection models[17, 18] are applied in the present research for the visualization and subsequent analysis of attack data collected by a network of honeypots, also known as a honeynet.

A honeypot has no authorized function or productive value within the corporate network other than to be explored, attacked or compromised.[19] Thus, a honeypot should not receive any traffic at all. Any connection attempt with a honeypot is then an attack or attempt to compromise the device or services that it is offering- is by default illegitimate traffic. From the security point of view, there is a great deal that may be learnt from a honeypot about a hacker's tools and methods in order to improve the protection of information systems.

In a honeynet, all the traffic received by the sensors is suspicious by default. Thus every packet should be considered as an attack or at least as a piece of a multi-step attack. Numerous studies propose the use of honeypots to detect automatic large scale attacks; honeyd[20] and nepenthes[21] among others. The first Internet traffic monitors known as Network Telescopes, Black Holes or Internet Sinks were presented by Moore *et al*. [22]

The remaining five sections of this study are structured as follows: section 2 briefly describes the topic of computer and network security (mainly Intrusion Detection). Section 3 presents the novel approach proposed for ID while the neural projection and visualization techniques applied in this research are described in section 4. Some experimental results for two different real-life datasets are then presented and comprehensively described in section 5. Finally, the conclusions of this interdisciplinary study and the future research lines are discussed in section 6. Additionally, Appendix A comprises a compendium of the remaining visualizations (not shown in previous sections) for the two analyzed datasets.

2 Computer and Network Security

This section introduces the main concepts of computer and network security that are the foundations of this study.

2.1 Intrusion Detection Systems

Intrusions can be produced by attackers that access the system, by authorized users that attempt to obtain unauthorized privileges, or by authorized users that misuse the privileges given to them. The complexity of such situations increases in the case of distributed network-based systems and insecure networks. When attackers try to access a system through external networks such as the Internet, one or several hosts may be involved. From a victim's perspective, intrusions are characterized by their manifestations, which might or might not include damage. [23] Some attacks may produce no manifestations while some apparent manifestations can be produced by system or network malfunctions.

An IDS can be defined as a piece of software that runs on a host, which monitors the activities of users and programs on the same host and/or the traffic on networks to which that host is connected.[24] The main purpose of an IDS is to alert the system administrator to any suspicious and possibly intrusive event taking place in the system that is being analyzed. Thus, they are designed to monitor and to analyze computer and/or network events in order to detect suspect patterns that may relate to a system or network intrusion.

Ever since the first studies in this field in the 80s,[3, 25] the accurate detection in real-time of computer and network system intrusions has always been an interesting and intriguing problem for system administrators and information security researchers. It may be attributed on the whole to the dynamic nature of systems and networks, the creativity of attackers, the wide range of computer hardware and operating systems and so on. Such complexity arises when dealing with distributed network-based systems and insecure networks such as the Internet.[26]

A standard characterization of IDSs, based on their detection method, or model of intrusions, defines the following paradigms:

- **Anomaly-based ID** (also known as behaviour-based ID): the IDS detects intrusions by looking for activity that differs from the previously defined "normal" behaviour of users and/or systems. In keeping with this idea, the observed activity is compared against "predefined" profiles of expected normal usage. It is assumed that all intrusive activities are necessarily anomalous. In real-life environments, instead of their being identical, the set of intrusive activities only intersects the set of anomalous activities in some cases. As a

consequence,[27] anomalous activities that are not intrusive are flagged as intrusive (i.e. false positives) and intrusive activities that are not anomalous are not flagged up (i.e. false negatives). Anomaly-based IDSs can support detection of novel (zero-day) attack strategies but may suffer from a relatively high rate of false positives.[28]

- **Misuse-based ID** (also known as knowledge-based ID): intrusions are detected by checking activity that corresponds to known intrusion techniques (signatures) or system vulnerabilities. Misuse-based IDSs are therefore commonly known as signature-based IDSs. They detect intrusions by exploiting the available knowledge on specific attacks and vulnerabilities. As opposed to anomaly detection, misuse detection assumes that each intrusive activity can be represented by a unique pattern or signature.[29] This approach entails one main problem; intrusions whose signatures are not archived by the system can not be detected. As a consequence, a misuse-based IDS will never detect a 0-day attack.[29] The completeness of such IDSs requires regular updating of their knowledge of attacks.
- **Specification-based ID**: it relies on program behavioural specifications reflecting system policies that are used as a basis to detect attacks.[30]

2.1.1 Snort

Snort, a libpcap-based[31] lightweight network intrusion detection system, is one of the most widely deployed IDS. It is a network-based, misuse-based IDS. Snort detects many types of malicious activity in the packet payload that can be characterized in a unique detection signature. It is focused on collecting packets as quickly as possible and processing them in the Snort detection engine. It is composed of three primary modules: a packet decoder, a detection engine and a logging and alerting subsystem.

Even if the capabilities of Snort allow a deep analysis of the traffic flows, what interests in this research is the detection, alerting and logging of the network packets as they arrive to the Honeynet system. Snort is used as a network data classifier, without discarding any packet. In that sense, in addition to the default rules of the Snort community, three basic rules that log all TCP, UDP and ICMP traffic are included, as shown in Table 1.

Table 1. Snort rules to log all TCP, UDP and ICMP traffic.

alert tcp $EXTERNAL_NET any ->$HOME_NET any (msg:"TCP"; sid:1000001;)
alert udp $EXTERNAL_NET any ->$HOME_NET any (msg:"UDP"; sid:1000002;)
alert icmp $EXTERNAL_NET any ->$HOME_NET any (msg:"ICMP"; sid:1000003;)

On the other hand, each incoming packet is inspected and compared with the default rule base. This way, besides alerting when the packet matches the three signatures shown above, many of them also match the Snort rule base signatures. Thereby, even if a big amount of packets cause more than one alarm to be triggered, it facilitates a simple way to separate the alarm set into two subsets:

- Alarms that have been triggered when matching the Snort default rule base. This dataset can be considered as known attack data.
- Alarms that did not match any of the known attack rules. Considered as the unknown data.

These two subsets will allow researchers to distinguish between the known and unknown traffic. This permits testing the success rate of Snort, and also visualizing the unknown traffic looking for new and unknown attacks. A clear advantage of using Snort IDS on this experiment is the ease of use, configuration and development of new rules.

2.2 Honeypots and Honeynets

The last few years, two monitoring systems for automatic large-scale attack detection have been proposed: honeypots and network telescopes. Since 1992 honeypots are used to deceive attackers to learn from the new attacks they accomplish.[32, 33, 34, 35] A honeypot is a decoy system consisting of some vulnerable computing resource used to distract attackers, as an early warning system for new attack proliferations and to ease the later analysis of the attacks (forensics).[36] When used to monitor activities derived from automatic attacks based on random or pseudo-random scanning, these systems have certain particularities. Unassigned IP addresses are given to these honeypots so every time a honeypot receives a connection request, it will be considered as suspicious. Nevertheless, the interaction level of the honeypot is fundamental. The higher the interaction between the honeypot and the attacker (response to TCP connection request for example), the more information can be

gathered and therefore a higher knowledge about the attack will be obtained. A system with a low level of interaction will also be valid to analyze the noise level, detect infected hosts, etc.

One of the most extended classifications of honeypots takes into account their level of interaction. Low interaction honeypots offer limited interaction with attackers and the most common ones only simulate services and operating systems. High interaction honeypots follow a different strategy: instead of using simulated services and operating systems, real systems and applications are used, usually running in virtual machines.

Somewhere between the two ones are medium interaction honeypots, which also emulate vulnerable services, but leave the operating system to manage the connections with their network protocol stack. Recently, a new type of honeypot has been proposed as a response to the behavioural change observed in the attackers. Instead of waiting for the attackers to reach traditional honeypots, client side honeypots, also known as honeyclients, scan communication channels looking for malware.

This study, based on the analyzed case studies, is focused on medium interaction honeypots.

In a honeynet, all the traffic received by the sensors is suspicious by default. Thus every packet should be considered as an attack or at least as a piece of a multi-step attack. Different platforms exist as observatories of malicious threats using honeypots. Examples are NoAH[20], and SGNET[21]. In this research, we are following this approach, based on the application of unsupervised learning to network level attacks collected from a honeypot-based observatory.

3 A Visualization-based Approach for Data Monitored by Honeypots

This study proposes the application of projection models for the visualization of traffic network attack data obtained by honeypots. There exist different approaches to collect attacks on the Internet, but there is still a lack of techniques that ease the comprehension and analysis of the information gathered. Visualization techniques have been applied to massive datasets for many years. These techniques are considered a viable approach to information seeking, as humans are able to recognize different features and to detect anomalies by inspecting graphs.[37] The underlying operational assumption of the proposed approach is mainly grounded in the ability to render the high-dimensional traffic data in a consistent yet low-dimensional representation. So, security visualization tools have to map high-dimensional feature data into a low-dimensional space for presentation. One of the main assumptions of the research presented in this study is that neural projection models will prove themselves to be satisfactory for the purpose of security data visualization through dimensionality reduction, analyzing complex high-dimensional datasets obtained by honeypots.

This problem of identifying patterns that exist across dimensional boundaries in high dimensional datasets is a challenging task. Such patterns may become visible if changes are made to the spatial coordinates. However, an *a priori* decision as to which parameters will reveal most patterns requires prior knowledge of unknown patterns.

Projection methods project high-dimensional data points onto a lower dimensional space in order to identify "interesting" directions in terms of any specific index or projection. Having identified the most interesting projections, the data are then projected onto a lower dimensional subspace plotted in two or three dimensions, which makes it possible to examine the structure with the naked eye. Projection methods can be smart compression tools that map raw, high-dimensional data onto two or three dimensional spaces for subsequent graphical display. By doing so, the structure that is identified through a multivariable dataset may be visually analyzed with greater ease.

Visualization tools can therefore support security tasks in the following way:

- Visualization tools may be understood intuitively (even by inexperienced staff) and require less configuration time than more conventional tools.
- Providing an intuitive visualization of data allows inexperienced security staff to learn more about standard network behaviour, which is a key issue in ID.[38] The monitoring task can be then assigned to less experienced security staff.
- As stated in [6] "*visualizations that depict patterns in massive amounts of data, and methods for interacting with those visualizations can help analysts prepare for unforeseen events*". Hence, such tools can also be used in security training.
- They can work in unison with some other security tools in a complementary way.

As with other machine learning paradigms, an interesting facet of ANN learning is not just that the input patterns may be precisely learned/classified/identified, but that this learning can be generalized. Whereas learning takes place within a set of training patterns, an important property of the learning process is that the network can generalize its results on a set of test patterns that were not previously learnt. Also, their capability to identify unknown patterns fits the 0-day attack[28] detection.

Due to the aforementioned reasons, the present work approaches the analysis of attack data from a visualization standpoint. That is, some neural projection techniques are applied for the visualization of data monitored by honeypots.

3.1 Previous Work

Great effort has been devoted to the ID field up to now, but several issues concerning IDS design, development, and performance are still open for further research.

Scant attention has been given to visualization in the ID field,[39] although visual presentations do, in general, help operators and security managers, in particular, to interpret large quantities of data. Most IDSs do not provide any way of viewing information other than through lists, aggregates, or trends of raw data. They can generate different alarms when an anomalous situation is detected, broaden monitoring tasks, and increase situational awareness. However, they can neither provide a general overview of what is happening in the network nor support a detailed packet-level inspection[7] as is the case under honeypots and honeynets.

Some other authors have previously addressed the analysis of traffic data and intrusion detection under the application of ANN[40, 41] and more in particular projection methods.[14, 42]

The underlying idea in this ongoing research is not only to detect anomalous situations under data sets monitored by honeypots but also to visualize protocol interactions and traffic volume. Packet-based ID, that is actually performed in this present research, has several advantages. [43]

Some Exploratory Projection Pursuit (EPP)[53] models have been previously applied to the ID field as part of a hybrid intelligent IDS[14, 15, 16]. Differentiating from previous studies, EPP models, are applied in the present study as a complementary tool to IDSs for the first time to analyze real complex high-dimensional honeynet data sets. In this sense, now the output of both the neural model and Snort (the novel applied IDS) are combined, together with some other customized visualizations for comprehensive analysis and understanding of network status.

4 Neural Visualization Techniques

The different projection models applied in this study are described in the following sections.

4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical model, introduced in [44] and independently in [45], that describes the variation in a set of multivariate data in terms of a set of uncorrelated variables each, of which is a linear combination of the original variables.

Its goal is to derive new variables, in decreasing order of importance, that are linear combinations of the original variables and are uncorrelated with each other. From a geometrical point of view, this goal mainly consists of a rotation of the axes of the original coordinate system to a new set of orthogonal axes that are ordered in terms of the amount of variance of the original data they account for. The optimal projection given by PCA from an N-dimensional to an M-dimensional space is the subspace spanned by the M eigenvectors with the largest eigenvalues.

According to [46], it is possible to describe PCA as a mapping of vectors $\mathbf{x^d}$ in an N-dimensional input space $(x_1,...,x_N)$ onto vectors $\mathbf{y^d}$ in an M-dimensional output space $(y_1,...,y_M)$, where $M \leq N$. $\mathbf{x}$ may be represented as a linear combination of a set of N orthonormal vectors W_i:

$$\mathbf{x} = \sum_{i=1}^{N} y_i W_i \tag{1}$$

Vectors W_i satisfy the orthonormality relation:

$$W_i^t W_j = \delta_{ij} \tag{2}$$

where δ_{ij} is the Kronecker delta.

Making use of equation (1), the coefficients y_i may be given by

$$y_i = W_i^T \mathbf{x} \tag{3}$$

which can be regarded as a simple rotation of the co-ordinate system from the original $\mathbf{x}$ values to a new set

of co-ordinates given by the $\mathbf{y}$ values. If only one subset $M < N$ of the basis vectors, W_i, is retained so that only M coefficients y_i are used, and having replaced the remaining coefficients by constants b_i, then each $\mathbf{x}$ vector may be approximated by the following expression:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{M} y_i W_i + \sum_{i=M+1}^{N} b_i W_i \qquad (4)$$

Consider the whole dataset of D vectors, $\mathbf{x}^{\mathbf{d}}$ where $d = 1,..., D$.

PCA can be performed by means of ANNs or connectionist models such as [47, 48, 49, 50, 51]. It should be noted that even if we are able to characterize the data with a few variables, it does not follow that an interpretation will ensue.

4.2 Cooperative Maximum Likelihood Hebbian Learning

The Cooperative Maximum Likelihood Hebbian Learning (CMLHL) model[52] extends the Maximum Likelihood Hebbian Learning (MLHL)[17] model, which is based on EPP. The statistical method of EPP was designed for solving the complex problem of identifying structure in high dimensional data by projecting it onto a lower dimensional subspace in which its structure is searched for by eye. To that end, an "index" must be defined to measure the varying degrees of interest associated with each projection. Subsequently, the data is transformed by maximizing the index and the associated interest. From a statistical point of view the most interesting directions are those that are as non-Gaussian as possible.

The MLHL model is based on the Negative Feedback Network and it associates an input vector, $\boldsymbol{x} \in \Re^D$, with an output vector, $\boldsymbol{y} \in \Re^Q$. In this case, the output of the network ($\boldsymbol{y}$) is computed as:

$$y_i = \sum_{j=1}^{\mathbf{N}} W_{ij} x_j, \forall i \qquad (5)$$

where, W_{ij} is the weight linking input j to output i.

Once the output of the network has been calculated, the activation (e_j) is fed back through the same weights and subtracted from the input:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i, \forall j \qquad (6)$$

Finally, the learning rule determines the way in which the weights are updated:

$$\Delta W_{ij} = \eta . y_i . sign(e_j) | e_j |^{p-1} \qquad (7)$$

where, η is the learning rate and p is a parameter related to the energy function.

The main difference between the basic MLHL model and its Cooperative version is the introduction of lateral connections.[52, 54, 55] After the Feed forward step (Eq. 5) and before the Feed back step (Eq. 6), lateral connections between the output neurons are applied as follows:

$$y_i(t+1) = [y_i(t) + \tau(b - Ay)]^+ \qquad (8)$$

where, τ is the "strength" of the lateral connections, b is the bias parameter and A is a symmetric matrix used to modify the response to the data. Its effect is based on the relation between the distances among the output neurons.

4.3 Curvilinear Component Analysis

Curvilinear Component Analysis (CCA)[56] is a nonlinear dimensionality reduction method. Developed as an improvement on the SOM, it tries to circumvent the limitations inherent in previous linear models such as PCA.

The principle of CCA is a self-organized neural network performing two tasks: a vector quantization of the submanifold in the data set (input space) and a nonlinear projection of these quantising vectors toward an output space, providing a revealing view of the way in which the submanifold unfolds. Quantization and nonlinear mapping are separately performed by two layers of connections: firstly, the input vectors are forced to become prototypes of the distribution using a vector quantization (VQ) method; then, the output layer builds a nonlinear mapping of the input vectors by considering Euclidean distances.

In the vector quantization step, the input vectors (x_i) are forced to become prototypes of the distribution by using competitive learning and the regularization method[57] of vector quantization. Thus, this step, which is intended to reveal the submanifold of the distribution, regularly quantizes the space covered by the data, regardless of the density. Euclidean distances between these input vectors $(X_{ij} = d(x_i, x_j))$ are considered, as the output layer has to build a nonlinear mapping of

the input vectors. The corresponding distances in the output space are also used $\left(Y_{ij} = d\left(y_i, y_j\right)\right)$.
Perfect matching is not possible at all scales when the manifold is "unfolding", so a weighting function $\left(F\left(Y_{ij}, \lambda_y\right)\right)$ is introduced, yielding the quadratic cost function:

$$E = \frac{1}{2}\sum_i \sum_{j \neq i} \left(X_{ij} - Y_{ij}\right)^2 F\left(Y_{ij}, \lambda_y\right) \qquad (9)$$

Where: λ is a user-tuned parameter allowing an interactive selection of the scale at which the unfolding takes place.
As regards its goal, the projection part of CCA is similar to other nonlinear mapping methods, in that it minimizes a cost function based on interpoint distances in both input and output spaces. Instead of moving one of the output vectors (y_i) according to the sum of the influences of every other y_j (as would be the case for a stochastic gradient descent), CCA proposes pinning down one of the output vectors (y_i) "temporarily", and moving all the other y_j around, disregarding any interactions between them. Accordingly, the proposed "learning" rule can be expressed as:

$$\Delta y_j = \alpha(t) F\left(Y_{ij}, \lambda_y\right)\left(X_{ij} - Y_{ij}\right)\frac{y_j - y_i}{Y_{ij}} \;\; \forall j \neq i \; (10)$$

Where: $\alpha(\;)$ is the step size that decreases over time.

4.4 Self Organizing Map

The Self-Organizing Map (SOM)[58] was developed as a visualization tool for representing high dimensional data on a low dimensional display. It is also based on the use of unsupervised learning. However, it is a topology preserving mapping model rather than a projection architecture.
The cerebral cortex of the human brain is one of the most complex biological systems. The different areas defined in this region are organized according to various sensory modalities: speech control, visual analysis, auditory control, etc. Each one of these areas consists of a large number of similar neurons that cooperate when carrying out their specific functions in which they have become specialized: auditory or hearing receptors, visualization, etc. Groups of neurons within each region respond jointly to excitations from the sensory cell they service. [59] There is a mapping of the features from sensory neurons to the associated spatial regions of the cortex. This biological feature mapping of the brain has been modelled reasonably well with ANNs. The computed SOMs are very similar to many brain maps as they also behave dynamically, in the same way, for example, as their magnification is adjusted in proportion to the occurrences of the stimuli. [60] Thus the SOM[61] is a proper example of artificial topology preserving maps, where closer neurons are activated by similar inputs or stimuli.
To mimic the biological brain maps, the SOM is composed of a discrete array of L nodes arranged on an N-dimensional lattice. These nodes are mapped into a D-dimensional data space while preserving their ordering. The dimensionality of the lattice (N) is normally smaller than that of the data, in order to perform the dimensionality reduction. The SOM can be viewed as a non-linear extension of PCA, where the global map manifold is a non-linear representation of the training data. [62]
Typically, the array of nodes is one or two-dimensional, with all nodes connected to the N inputs by an N-dimensional weight vector. The self-organization process is commonly implemented as an iterative on-line algorithm, although a batch version also exists. An input vector is presented to the network and a winning node, whose weight vector W_c is the closest (in terms of Euclidean distance) to the input, is chosen:

$$c = \arg\min_i\left(\left\|\mathbf{x} - W_i\right\|\right) \qquad (11)$$

The SOM is therefore a vector quantizer, and data vectors are quantised to the reference vector in the map that is closest to the input vector. The weights of the winning node and the nodes close to it are then updated to move closer to the input vector. There is also a learning rate parameter (η) that usually decreases as the training process progresses. The weight update rule is defined as:

$$\Delta W_i = \eta h_{ci}\left[\mathbf{x} - W_i\right], \forall i \in N^{(c)} \qquad (12)$$

When this algorithm is sufficiently iterated, the map self-organizes to produce a topology-preserving mapping of the lattice of weight vectors to the input space based on the statistics of the training data.
This neural model is applied here for comparative purposes as it is one of the most widely used unsupervised neural models for visualizing structure in high-dimensional data sets.

5 Experimental Study

Researchers usually make use of known attack datasets such as the well known DARPA dataset[63, 64, 65] or the KDD Cup '99 sub-dataset[66, 67] in order to validate their developed systems. However, these data are simulated, non-validated and irregular[52] so they are not fully reliable. Even if the results obtained by such systems are good, no one can assure that the applied algorithms will make the system more secure or will detect real attacks. This is the main reason of using two real attack data sets coming from a running honeynet in this research.

The experimental work has been done by using data related to five months of real attacks that reached the Euskalert network.[68] These data are depicted through different neural projection and visualization techniques in order to discover real attack behaviour and strategies.

The Euskalert project[68] has deployed a network of honeypots in the Basque Country (northern Spain) where eight companies and institutions have installed one of the project's sensors behind the firewalls of their corporate networks. The honeypot sensor transmits all the traffic received to a database via a secure communication channel. These partners can consult information relative to their sensor (after a login process) as well as general statistics in the project's website. Once a big amount of data has been collected, the information available can be used to analyze attacks received by the honeynet at network and application level.

Euskalert is a distributed honeypot network based on a Honeynet GenIII architecture.[53] The developed architecture of Euskalert is shown in Fig. 1. The various sensors installed in corporate networks of the different participants are shown in the left of Fig.1.

Every sensor has a permanently established an encrypted connection (using different virtual private networks, also known as VPN) to a tunnel server. The latter is in the DMZ (Demilitarized Zone) of Mondragon University. Any attack to one of the sensors is redirected through these tunnels to reach the Honeypot (right side of Fig. 1), which is the responsible for responding to any connection attempt. The traffic also passes through a server responsible for collecting all the information which is then displayed on the Web platform.[68]

This honeypot system has received about 164 packets a day on average. All the traffic is analyzed by the Snort IDS, and an alert is launched whenever the packet matches a known attack signature.

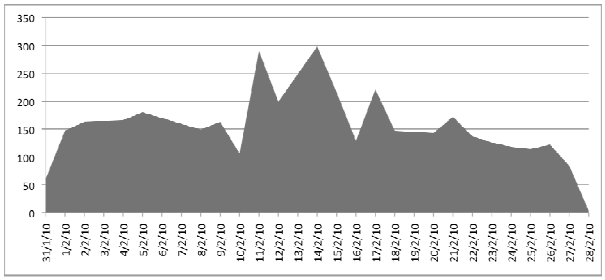

Fig. 2. Temporal distribution of the traffic volume in terms of number of packets captured by Euskalert during February 2010.

The following features were extracted from each one of the records in the dataset:

- **Time**: the time when the attack was detected. Difference in relation to the first attack in the dataset (in minutes).
- **Protocol**: either TCP, UDP or ICMP (codified as three binary features).
- **Ip_len**: number of bytes in the packet.
- **Source Port**: number of the port from which the source host sent the packet. In the case of the ICMP protocol, this represents the ICMP type field.
- **Destination Port**: destination host port number to which the packet was sent. In the case of the ICMP protocol, this represents the ICMP type field.
- **Flags**: Control bits of a TCP packet, which contains 8 1 bit values.

Two different real-life case studies are analyzed in this research as attack behavior may change in time. First, a snapshot of one-month data (February 2010) is analyzed to observe its internal structure. A five-month period (February-June 2010) is taken later to analyze how attacks have changed and new trends are discovered.

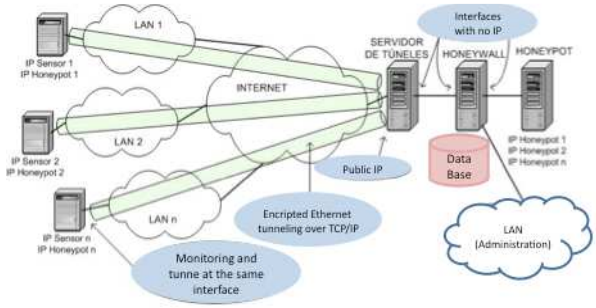

Fig. 1. Architecture of the Euskalert network.

The previously introduced projection models have been applied to these two case studies, whose results are shown and described in the following sections.

5.1 Case study 1: a 1-month dataset

For this real case study, the logs coming from Euskalert and Snort have been gathered during one month (February 2010). Fig. 2 shows the traffic volume in terms of number of packets received for that period of time.

The February 2010 dataset contains a total of 3798 packets, including TCP, UDP and ICMP traffic received by the distributed honeypot sensors. The characterization of the traffic in the dataset is shown in Table 2. The table shows which alerts have been triggered in that period of time and their percentage. Those signatures starting with "Wormledge" are automatically generated and not present in the default signature database.

Table 2. Characterization of traffic data captured by Euskalert, during February, 2010.

Signature	# Packets	%
Unknown Traffic	3404	89,62
BLEEDING-EDGE POLICY Reserved IP Space Traffic - Bogon Nets 2	127	3,34
BLEEDING-EDGE WORM Allaple ICMP Sweep Ping Inbound	58	1,52
ICMP PING	75	1,97
Wormledge, microsoft-ds, smb directory packet (port 445). SMBr...PC NETWORK PROGRAM 1.0...LANMAN1.0...Windows for Workgroups 3.1a...LM1.2X002...LANMAN2.1...NT LM 0.12 . Created on 2007-08-07	34	0,89
Wormledge, KRPC Protocol (Kademlia RPC), BitTorrent information exchange:ping query. Created on 2007-08-07	11	0,28
Wormledge, NetBios Session Service (port 139). Payload CKFDENECFDEFFCFGAAAAAAAAAAAAAAAA. Created on 2007-08-07	7	0,18
Wormledge, NetBios Name Query (udp port 137). Payload CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA. Created on 2007-08-07	7	0,18
Wormledge, Microsoft RPC Service, dce endpoint resoluction (port 135). Created on 2007-08-07	7	0,18
WEB-IIS view source via translate header	6	0,15
BLEEDING-EDGE SCAN LibSSH Based SSH Connection - Often used as a BruteForce Tool	5	0,13

From this dataset, it may be said that a misuse detection-based IDS such as Snort is only capable of identifying about 10.38% of bad-intentioned traffic. Furthermore, it was demonstrated that only 2% of the unsolicited traffic was identified by the IDS when automatically generated signatures were included from a previous work.[69] Thus, a deeper analysis of the data is needed in order to discover the internal structure of the remaining 90% of the traffic. Explaining the behaviour of the unknown traffic is a difficult task that must be performed to better protect computer networks and systems. In order to obtain more knowledge, several neural projection models have been applied and the results and conclusions obtained are shown in the following sub-sections.

In the visualizations obtained, the data are depicted with different colors and shapes, taking into account the different original features of the data. In the shown projection, the axes are combinations of the features contained in the original datasets. Then, the X and Y axes of the projections can not be associated to a unique original feature.

5.1.1 CMLHL Projections

The CMLHL-training parameter values for the projections in this section were: number of iterations = 10,000, learning rate = 0.0208, p parameter = 2.1429, and τ parameter = 0.067.

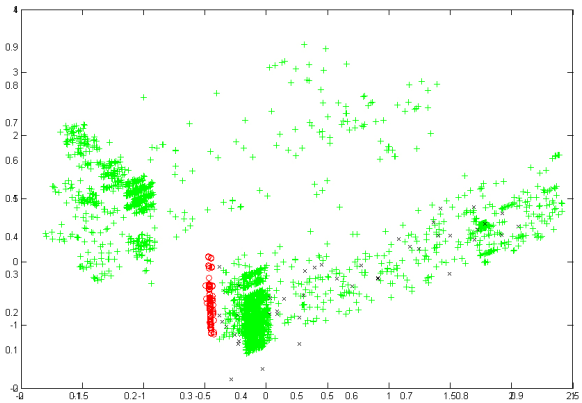

Fig. 5. CMLHL projection of 1-month data - Protocol.

Fig. 3 shows the CMLHL projection by considering the output generated by Snort. Packets that triggered any alarm are depicted as black crosses while packets that were not identified as anomalous are depicted as red circles.

After analyzing this projection (Fig. 3), we prove the poor detection performance of Snort IDS when filtering honeypot traffic. CMLHL provides a way of differentiating known from unknown traffic at a naked eye. Most of the traffic corresponds to unknown packets, or at list to traffic that Snort is not capable of identifying using all of its predefined rule sets.

Fig. 4 shows the CMLHL projection by considering the time (in minutes) to depict the packets; from 0 to 6692: red circles, from 6693 to 13384: black crosses, from 13385 to 20076: green pluses, from 20077 to 26768: magenta stars, from 26769 to 33460: yellow squares, and from 33461 to 40148: cyan diamonds.

The temporal evolution on that month shows that same traffic patterns repeat over time, as almost every cluster have similar shapes. This happens with both known and unknown traffic (shown in Fig. 3). It can be concluded

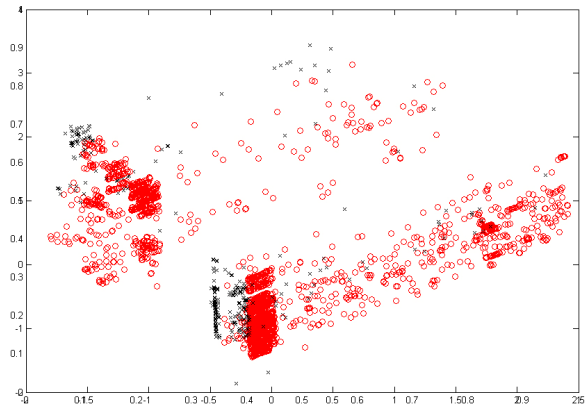

Fig. 3. CMLHL projection of 1-month data - Snort output.

that anomalous or unknown behavior is not a one off event, but a recurring pattern in time instead.

Fig. 5 shows the CMLHL projection by considering the protocol to depict the packets; ICMP: red circles, UDP: black crosses, TCP: green pluses.

After analyzing this projection (Fig. 5), it is easy to observe that most of the attacks collected target TCP protocol. This is a logical conclusion as most attacks target those kind of services, like Microsoft's netBios for example. It also brings the attention the fact that most of the ICMP traffic (red circles) belongs to the Snort's known traffic seen in Fig. 3.

Fig. 6 shows the CMLHL projection by considering the IP length (in bits) to depict the packets; from 28 to 273: red circles, from 274 to 519: black crosses, from 520 to 765: green pluses, from 766 to 1011: magenta stars, from 1012 to 1257: yellow squares, and from 1258 to 1500: cyan diamonds.

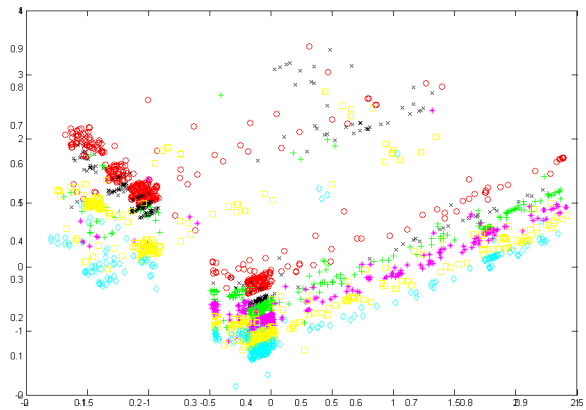

Fig. 4. CMLHL projection of 1-month data - Time.

Most of the traffic is composed of small packets, but it can also be observed very large packets received by the honeypot sensors. These are synonym of receiving malware, vulnerability exploits, or DoS attacks.

Fig. 7 shows the CMLHL projection by considering the source port to depict the packets; from 3 to 10903: red circles, from 10904 to 21803: black crosses, from 21804 to 32703: green pluses, from 32704 to 43603: magenta stars, from 43604 to 54503: yellow squares, and from 54504 to 65401: cyan diamonds.

Fig. 8 shows the CMLHL projection by considering the destination port to depict the packets; from 3 to 10371: red circles, from 10371 to 20739: black crosses, from 20739 to 31107: green pluses, from 31107 to 41475: magenta stars, from 41475 to 51843: yellow squares, and from 51843 to 62205: cyan diamonds.

Projections for source and destination ports show an obvious observation for the packets. Automatic categorization of the features groups most of the packets according to port numbers varying from 3 to 10371 and 10903. Source port of received traffic should be bigger than 1023 (non privileged ports). Destination port, or the port where known services listen for new connections are usually under 1023 (privileged ports). This is why source port is quite uniformly distributed in Fig. 7, and destination port has a big prevalence on the red cluster. Even though, we still see clusters with destination ports above 10371. This may be a side effect of DoS attacks, also known as backscatter. The term backscatter refers to unsolicited traffic that is the result of responses to attacks spoofed with a network's IP address.[70] For example, when an attacker launches a DoS attack against a victim, he usually spoofs it's own IP address with another one. When this spoofed IP address matches one of Euskalert's sensors addresses,

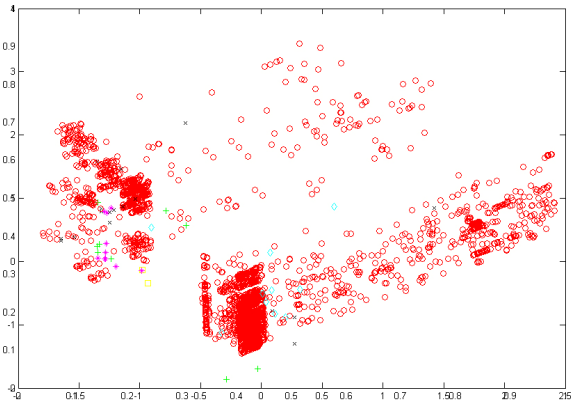

Fig. 6. CMLHL projection of 1-month data - IP length.

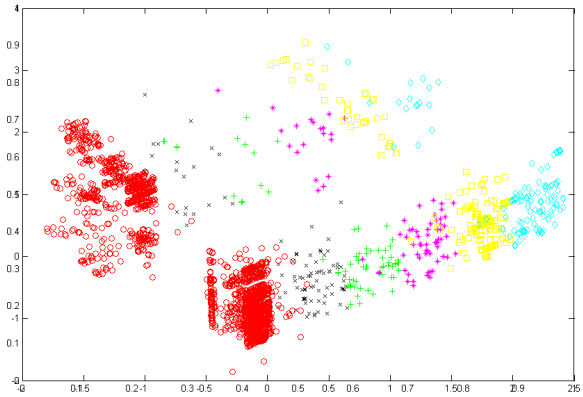

Fig. 7. CMLHL projection of 1-month data - Source port.

the response will be captured.

Finally, the flags were considered to depict the packets (Fig. 9); from 0 to 5: red circles, from 6 to 11: black crosses, from 12 to 17: green pluses, from 18 to 23: magenta stars, and from 24 to 25: yellow squares.

Value for flags in both UDP and ICMP traffic is considered as a 0 (red circles). This gives interesting results because it distinguishes very clearly the different types of packets.

Continuing with DoS attacks, the flag bits of the TCP header gives valuable information for analyzing this phenomenon. Every TCP packet has a 6 bit length control field, consisting of:

- URG: Urgent Pointer
- ACK: Acknowledgment
- PSH: Push Function
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: No more data from sender

According to the normal behavior of TCP, any

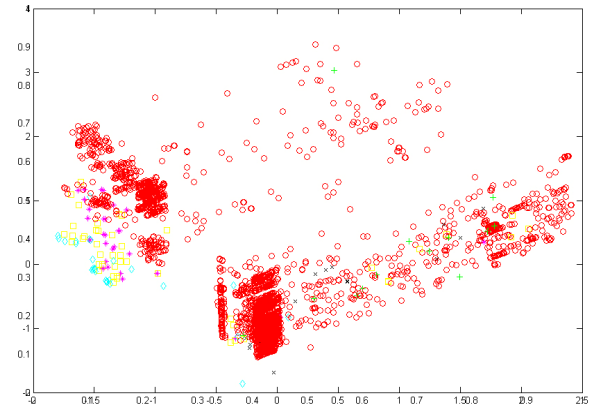

Fig. 8. CMLHL projection of 1-month data - Destination port.

computer that remains passive or does not initialize any connection can expect those combinations of control bits: SYN, ACK, PSH, FIN. Consequently, any other control bit combination will indicate the reception of DoS activity, at least as a side effect.
If we look at the different flag combinations found in this dataset, we find:

- No flags (0): corresponds to UDP and ICMP packets.
- SYN (2).
- RST (4).
- ACK (16).
- PSH+FIN (17).
- SYN+ACK (18).
- RST+ACK (20).
- PSH+ACK (24).
- PSH+ACK+FIN (25).

Being this said, it can be argued that packets containing SYN+ACK, RST, and RST+ACK flags activated are a sign of backscatter that honeypot sensors receive from the victim. According to Fig. 9 data is clearly structured, but the automatic categorization provided by CMLHL groups combinations of different flag bits into the same clusters. Being this considered, no concrete conclusions can be obtained, and a more meaningful categorization will be provided for the next case study, as the amount of data is bigger providing enough information for this issue.
All those visualizations are consequently in general very helpful information for explaining and helping security administrators to know the different traffic behavior that reach their systems.

5.1.2 Comparative Study

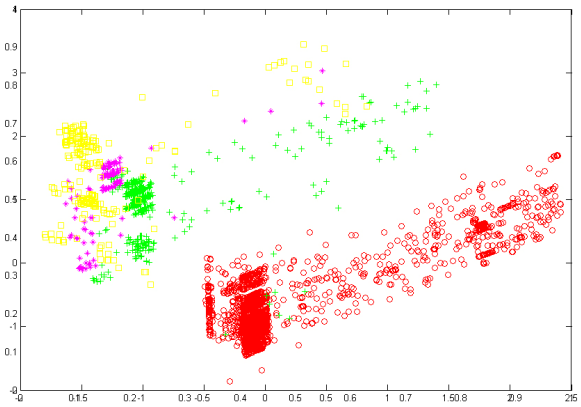

Fig. 9. CMLHL projection of 1-month data - Flags.

The CMLHL projections are compared with those of other dimensionality-reduction models (PCA, MLHL, CCA, and SOM). Several experiments were required to tune the SOM to different options and parameters: grid size, batch/online training, initialization, number of iterations and distance criterion, among others. In the case of CCA, other parameters, such as initialization, epochs and distance criterion were tuned. Only the best results (from the standpoint of the projection), which were obtained after tuning the models, are included in this work.
For the sake of brevity, only the best projections for each model are shown in this comparative study. The remaining visualizations are gathered in Appendix A.
The different ranges when visualizing a feature (Snort output, source/destination port, flags, protocol, etc.) are the same for PCA, MLHL and CCA.

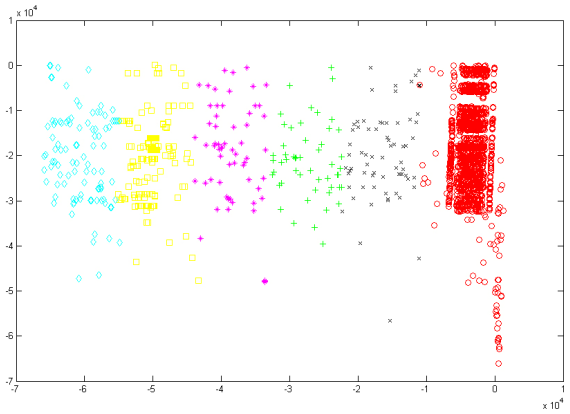

Fig. 10. PCA projection of 1-month data - Source port.

5.1.2.1 Principal Component Analysis

Fig. 10 shows the PCA projection of the case study 1 by using the source port, and Fig. 11 shows the PCA projection of the case study 1 by using the time feature. These two first principal components amounts to 84.33% of original data's variance.

Same security conclusions can be extracted from those projections; there is a very similar distribution of packets in time, and most of them have non-privileged ports as source port numbers.

5.1.2.2 Maximum Likelihood Hebbian Learning

Fig. 12 shows the MLHL projection of the case study 1 by using the destination port, while Fig. 13 shows the MLHL projection of the case study 1 by using the time feature. The MLHL-training parameter values for these two projections were: number of iterations = 10,000, learning rate = 0.0208, and *p* parameter = 2.1429.

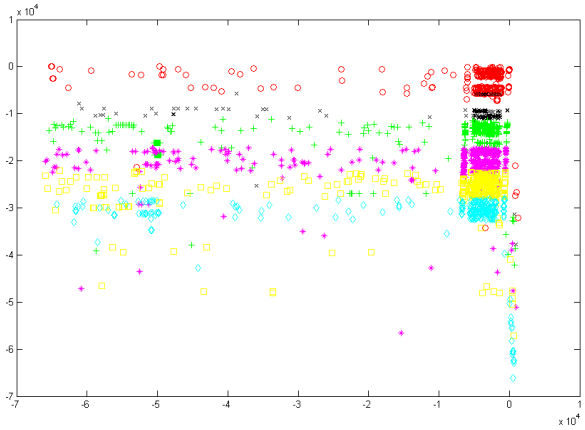

Fig. 11. PCA projection of 1-month data - Time.

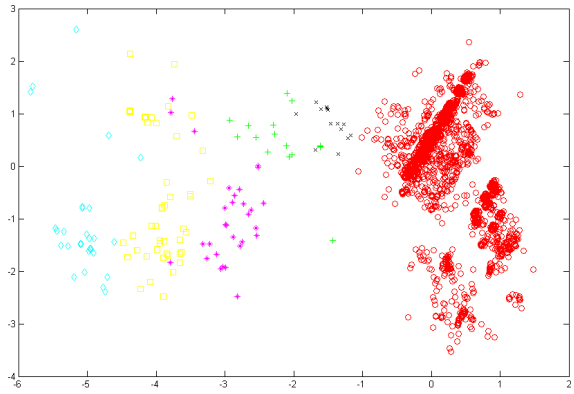

Fig. 12. MLHL projection of 1-month data - Destination port.

After analyzing these projections, it can be seen that projections using time information are less clear than others, while destination port can be clearly understood with the previous explanations given.

5.1.2.3 Curvilinear Component Analysis

Fig. 14 shows the CCA projection of the case study 1 by using the Snort output, and Fig. 15 shows the CCA projection of the case study 1 by using the protocol. Some parameters, such as alpha, lambda, number of epochs and distance criterion were tuned. The final selected parameter values were: standardized Euclidian distance, lambda = 230,000, alpha = 0.5 and 10 epochs.

Those projections show a visual explanation of the distribution of packets identified by Snort and those who are not, and if we analyze both of them as a whole, we observe that Snort could identify all of the ICMP

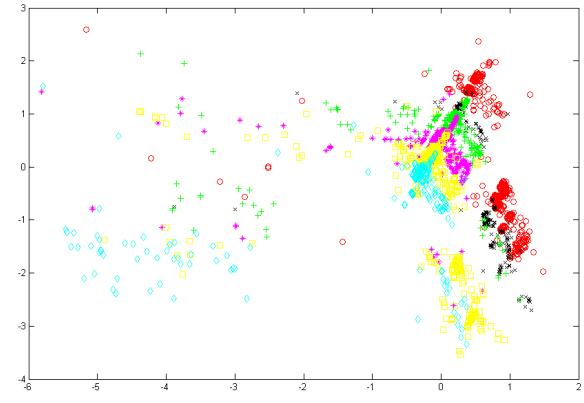

Fig. 13. MLHL projection of 1-month data - Time.

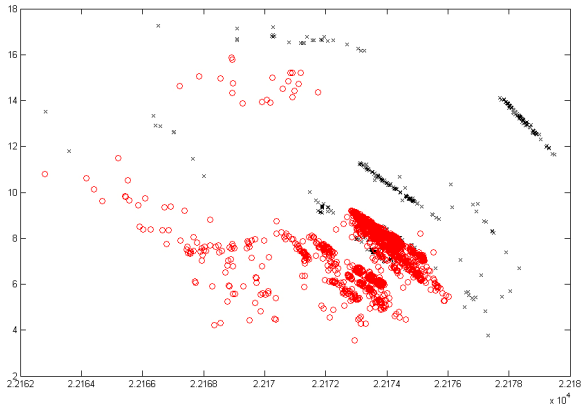

Fig. 14. CCA projection of 5-month data - Snort output.

packets in that month (red line in Fig. 15).

5.1.2.4 Self-Organizing Map

Finally, the SOM was also applied to the 1-month dataset. In order to analyze the resulting maps, the different ranges of each original feature (see section 5.1.1) were numbered.

Fig. 16 shows the U-matrix of the SOM mapping of the case study 1.

Fig. 17 shows the SOM map of the case study 1 by using the protocol; 1 represents ICMP, 3 UDP and 5 TCP. Fig. 18 shows the SOM map of the case study 1 by using the Snort output; 1 represents the packets identified as attacks while 0 represents the undetected attacks. For the SOM, the following options and parameters were tuned: grid size, batch/online training, initialization, number of iterations and distance criterion among others. The used parameter values were: linear

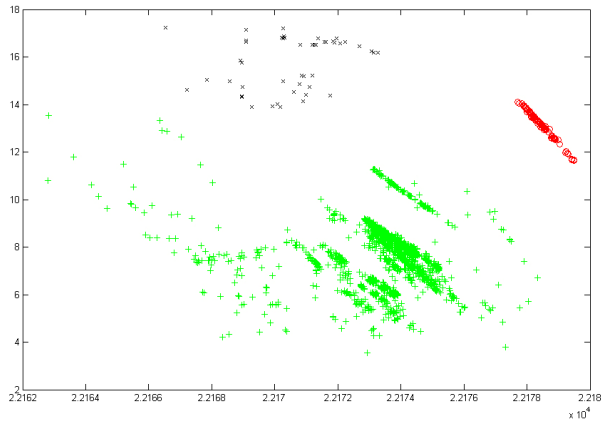

Fig. 15. CCA projection of 1-month data - Protocol.

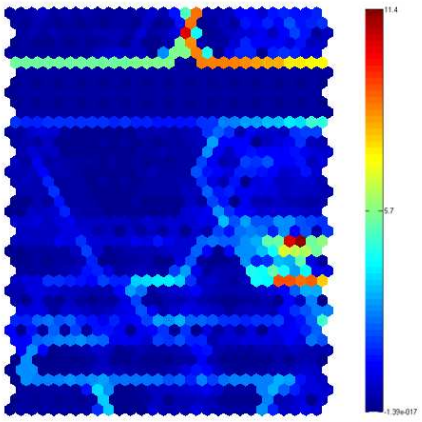

Fig. 16. SOM mapping of 1-month data - U-matrix.

initialization, batch training, hexagonal lattice, “Cut Gaussian” neighborhood function, and grid size (determined by means of a heuristic formula) = 15x21.

It can be easily seen in Fig. 17 that the SOM clusters the data in three big clusters. Each one of them contains all the traffic related with each one of the protocols (ICMP: upper left corner, UDP: upper right corner, and TCP: the rest of the mapping).

After analyzing the mapping in Fig. 18, it can be concluded that SOM is not able to cluster the data distinguishing the traffic classification of Snort (alarm/no alarm). The cluster in the upper left section is the only one identifying traffic of only one class (packets that triggered an alarm), while the other ones identify traffic of the two classes.

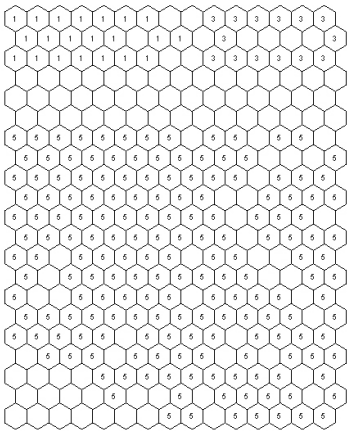

Fig. 17. SOM mapping of 1-month data - Protocol.

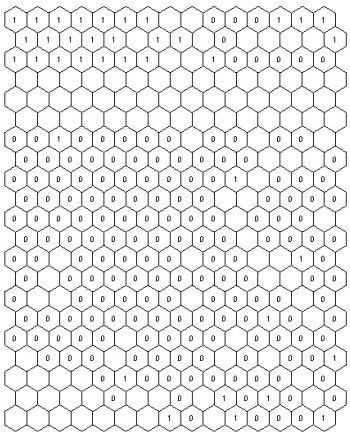

Fig. 18. SOM mapping of 1-month data - Snort output.

5.2 Case study 2: a 5-month dataset

For this experiment, it has been analyzed the logs coming from Euskalert and Snort gathered during five months starting from February 2010. Fig. 19 shows the traffic volume in terms of number of packets received for that period of time.

The dataset contains a total of 22601 packets, including TCP, UDP and ICMP traffic received by the distributed honeypot sensors. The characterization of the traffic in this dataset is shown in Table 3, which shows which alerts have been triggered in that period of time and their percentage. Those signatures starting with "Wormledge" are automatically generated and not present in the default Snort signature database. As it can be seen, the biggest group of signatures are those generated for unknown packets (both TCP, UDP and ICMP), and also the automatically generated signatures from a previous work.[69]

From this dataset, it may be said that a misuse detection-based IDS such as Snort is only capable of identifying less than 3,75% (847 packets out of 22601) of bad-intentioned traffic. Thus, a deeper analysis of the data is needed in order to discover the internal structure of the remaining 96,25% of the traffic data set. As in case study 1, neural unsupervised models are applied in order to explain the behavior of the unknown traffic.

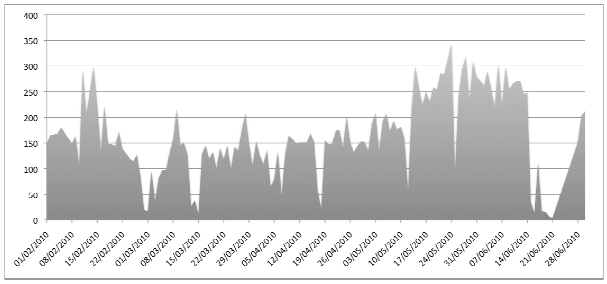

Fig. 19. Temporal distribution of the traffic volume in terms of number of packets captured by Euskalert from February to June, 2010.

Table 3. Characterization of traffic data captured by Euskalert, from February to June, 2010.

Signature	# Packets	%
Unknown TCP packet	19096	**84,49183664**
Reserved IP Space Traffic - Bogon Nets	1071	**4,738728375**
Unknown packet	741	**3,27861599**
Unknown UDP packet	397	**1,756559444**
ICMP ping	290	**1,283129065**
WORM Allaple ICMP Sweep Ping Inbound	251	**1,110570329**
Wormledge, KRPC Protocol (Kademlia RPC), BitTorrent information exchange:ping query	99	**0,438033715**
Wormledge, Slammer Worm	62	**0,274324145**
Wormledge, Microsoft-ds, smb directory packet (port 445)	62	**0,274324145**
Wormledge, MS-SQL-Service(port tcp 1433). Payload 1.4.3.3.O.D.B.C.	58	**0,256625813**
ICMP PING speedera	40	**0,176983319**
Wormledge, NetBios Name Query (udp port 137). Payload CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.	35	**0,154860404**
Wormledge, Possible SQL Snake/Spida Worm (port tcp 1433).	34	**0,150435821**
Wormledge, NetBios Session Service (port 139)	34	**0,150435821**
SIP TCP/IP message flooding directed to SIP proxy	33	**0,146011238**

5.2.1 CMLHL Projections

CMLHL was applied in order to analyze the dataset described above and to identify its inner structure.

Fig. 20 shows the CMLHL projection by considering the output generated by Snort. Packets that triggered any alarm are depicted as black crosses while packets that were not identified as anomalous are depicted as red circles.

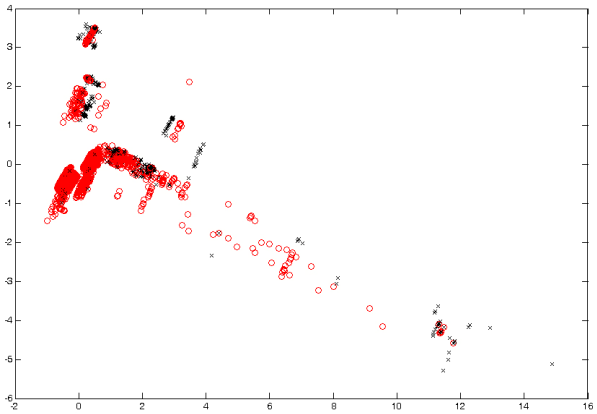

Fig. 20. CMLHL projection of 5-month data - Snort output.

Visualization of packets using Snort output shows the detection rate of the most used misuse-based IDS. Black crosses are the only ones detected by Snort, where all of the records constitute a suspicious activity by default. It is therefore important to use additional supporting systems, such as the visualization aids proposed in this study, to show a more comprehensive picture of what is actually happening and how an IDS is performing.

Fig. 21 shows the CMLHL projection by considering the time to depict the packets; from 0 to 27044: red circles, from 27045 to 54089: black crosses, from 54090 to 81134: green pluses, from 81135 to 108179: magenta stars, from 108180 to 135224: yellow squares, and from 135225 to 162267: cyan diamonds.

It can be observed that clusters are highly overlapped, which means that temporal distribution of attacks is very homogenous, so they constantly repeat over time. If we look some of the attacks at detail, we still see very

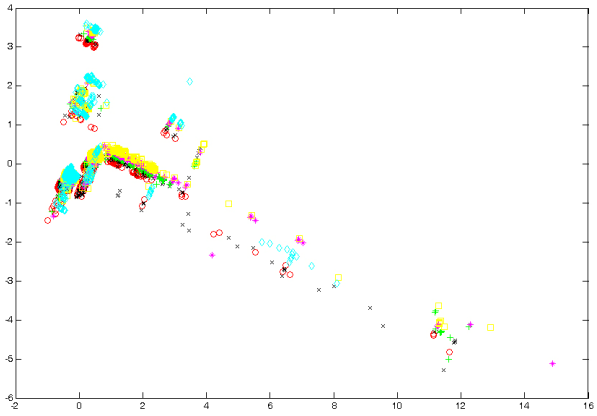

Fig. 21. CMLHL projection of 5-month data - Time.

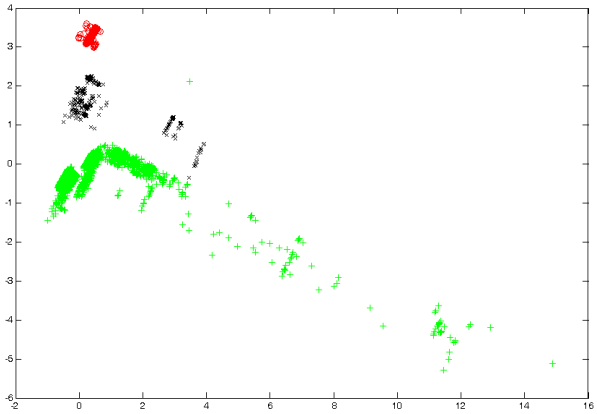

Fig. 22. CMLHL projection of 5-month data - Protocol.

old worm instances such as Slammer or Blaster (see Table 3).

Fig. 22 shows the CMLHL projection by considering the protocol to depict the packets; ICMP: red circles, UDP: black crosses, TCP: green pluses.

The visualization of data focusing on the protocol of the packets, shows the volume of attacks received by the honeypot sensors. Most of the attacks target TCP protocol, followed by UDP and ICMP. This information is not valuable by itself, but helps understanding some other situations that will be explained later.

Fig. 23 shows the CMLHL projection by considering the IP length to depict the packets; from 28 to 326: red circles, from 327 to 625: black crosses, from 626 to 924: green pluses, from 925 to 1223: magenta stars, from 1224 to 1522: yellow squares, and from 1523 to 1816: cyan diamonds.

The visualization of the length field data gives a valuable sight to analyze the traffic and focus on large

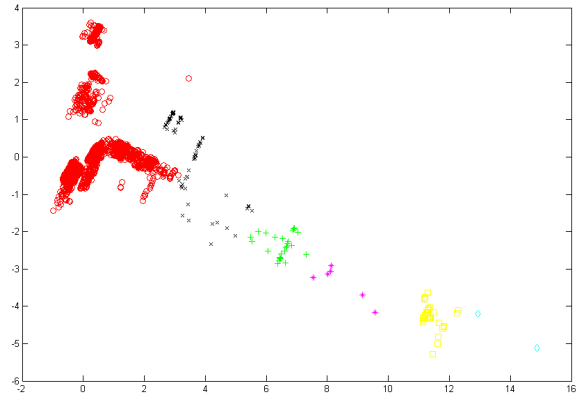

Fig. 23. CMLHL projection of 5-month data - IP length.

packets in order to identify both traffic sending malware and also traffic containing large payloads, usually trying either buffer overflow attacks or certain kinds of DoS attacks. Moreover, if we compare this graph with the protocol representation, it can be concluded that attacks with the biggest length belong to TCP, medium length belong to TCP and UDP, and smallest length to ICMP.
This assumption is coherent since ICMP traffic is composed of echo and reply packets in the dataset, often very small. In fact, if we see large ICMP packets we should consider them as Ping of Death type attacks (consisting of packets of 64 Kbyte size).
However UDP protocols typically use small packet size (i.e. DNS, etc.) and only tend to use large packets when used for Network File System protocol (NFS) for file sharing or for video or audio streaming. The latter is not possible in a honeynet, as it never starts streaming applications. We may found NFS packets, as vulnerabilities have been found in the past for that service.
Fig. 24 shows the CMLHL projection by considering the source port to depict the packets; from 3 to 10924: red circles, from 10925 to 21846: black crosses, from 21847 to 32768: green pluses, from 32769 to 43690: magenta stars, from 43691 to 54612: yellow squares, and from 54613 to 65529: cyan diamonds.
Fig. 25 shows the CMLHL projection by considering the destination port to depict the packets; from 3 to 10924: red circles, from 10925 to 21846: black crosses, from 21847 to 32768: green pluses, from 32769 to 43690: magenta stars, from 43691 to 54612: yellow squares, and from 54613 to 65534: cyan diamonds.
Once again, prevalence of red circles in Fig. 15 and Fig. 16 shows that most connections are established from a port under 10924 to a destination port in the same range.

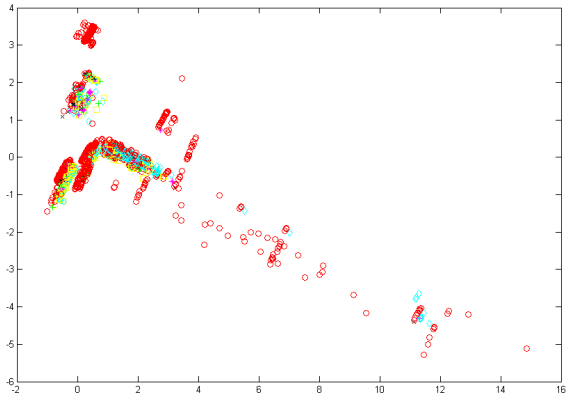

Fig. 24. CMLHL projection of 5-month data - Source port.

Having this said, a new categorization for both source

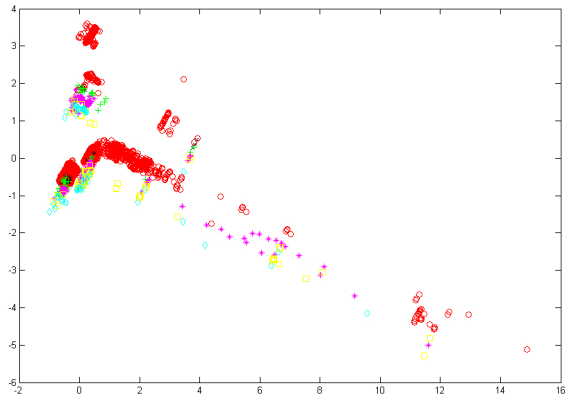

Fig. 25. CMLHL projection of 5-month data - Destination port.

and destination ports have been done. The new categorization helps on understanding the data from a logical viewpoint; source and destination ports can be naturally categorized into privileged and non-privileged ports. Doing this traffic patterns can be understood.
Finally, the flags were considered to depict the packets (Fig. 26); from 0 to 11: red circles, from 12 to 23: black crosses, from 24 to 35: green pluses, from 36 to 47: magenta stars, from 48 to 59: yellow squares, and from 60 to 68: cyan diamonds.
Fig. 26 results on a very useful visualization for analyzing backscatter. Once again default categorization of flag bit numbers does not permit a deeper analysis of the phenomenon.
For further analysis, some other visualizations of the CMLHL projection were generated. Those visualizations have new categories calculated for features source port, destination port and flags.
Fig. 27 shows the CMLHL projection by considering different ranges of the source port to depict the packets; from 3 and 8: red circles, from 0 to 1023 (excluding 3 and 8): black crosses, and from 1024 to 54612: green pluses. This new categorization shows a better perspective of what is happening on the honeypot. Red circles show ICMP packets, while green pluses exhibit the normal behavior for source port numbers in packets arriving to Euskalert. The honeypot acts as an application server, so different hosts in the Internet connect to the offered services using a source port above 1023. Instead, black crosses show source ports below 1023. The most provable explanation to this fact is, once again, backscatter. We are receiving responses of a spoofed probe, scan or DoS from a remote server in the Internet. This responses use a privileged number as

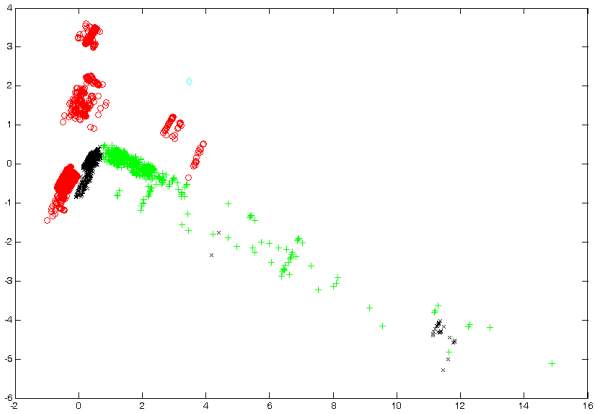

Fig. 26. CMLHL projection of 5-month data – Flags.

source port. We can observe these packets in combination with Fig. 20.

Fig. 28 shows the CMLHL projection by considering different ranges of the destination port to depict the packets; 3 and 8: red circles, from 0 to 1023 (excluding 3 and 8): black crosses, and from 1024 to 54612: green pluses.

This visualization shows that Euskalert receives packets and connection attempts to ports above 1023. We find two possible explanations of this observation. In one hand, there are attack attempts to applications listening on ports above 1023. In this case we should focus on this ports and if we find any prevalence then create a new simulated service for that application. In the other hand, we receive backscatter, being this port the source port of the attacker.

Fig. 29 shows the CMLHL projection by considering different ranges of the flags to depict the packets; 18 and 28: red circles, all the remaining values: black

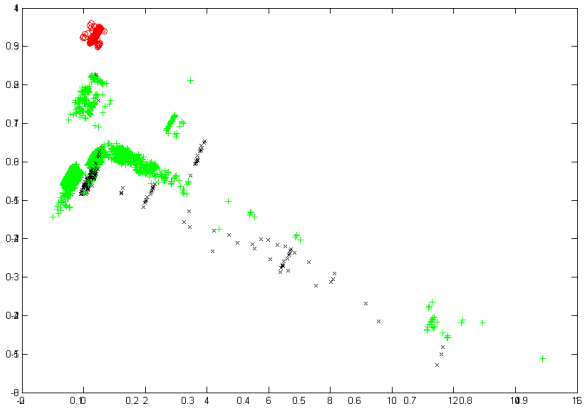

Fig. 27. CMLHL projection of 5-month data - Source Port v2.

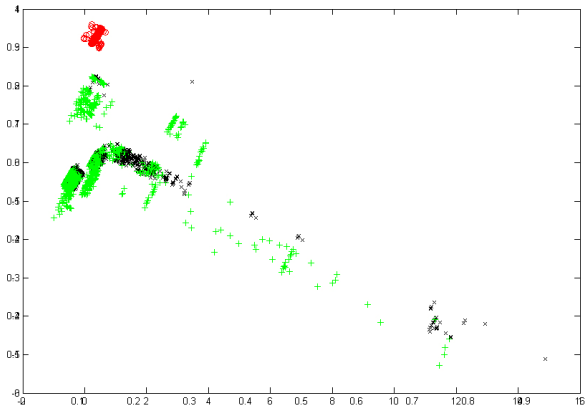

Fig. 28. CMLHL projection of 5-month data - Destination Port v2.

crosses.

Flags 18 and 28 correspond to SYN+ACK and RST+PSH+ACK flag combinations respectively. In combination with Fig. 18, it can be argued that these packets are related with responses from attack victims rather than from infected machines or attackers. Therefore we are capable of graphically detecting probes, scans or DoS attacks directed to remote servers.

5.2.2 Comparative Study

As in the case study 1, the CMLHL projections are compared with those of other dimensionality-reduction models (PCA, MLHL, CCA, and SOM).

5.2.2.1 Principal Component Analysis

Fig. 30 shows the PCA projection of the case study 2 by using the Snort output. Packets that triggered any alarm are depicted as black crosses while packets that were

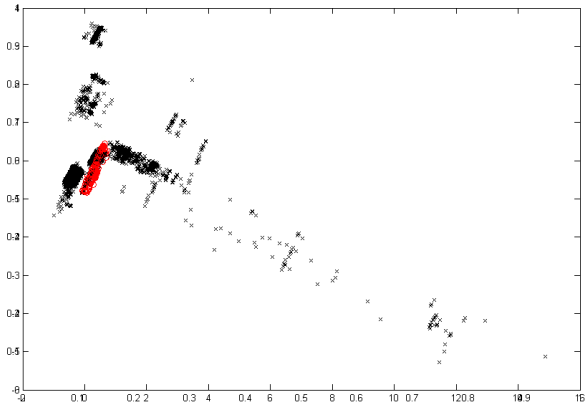

Fig. 29. CMLHL projection of 5-month data - Flags v2.

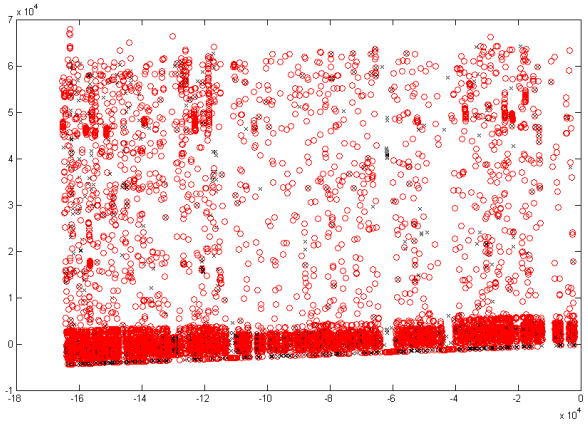

Fig. 30. PCA projection of 5-month data – Snort output.

not identified as anomalous are depicted as red circles.

Fig. 31 shows the PCA projection of the case study 2 by using the time feature to depict the packets; from 0 to 27044: red circles, from 27045 to 54089: black crosses, from 54090 to 81134: green pluses, from 81135 to 108179: magenta stars, from 108180 to 135224: yellow squares, and from 135225 to 162267: cyan diamonds. The two first principal components amount to 96.87% of original data's variance.

After analyzing these visualizations it can be said that Fig. 30 does not provides a clear idea of the situation as there is not a clear distinction of groups. On the contrary, Fig. 31 offers a clear representation of the observed conclusions, were the packet distribution in time is constant.

5.2.2.2 Maximum Likelihood Hebbian Learning

The MLHL-training parameter values for the projections in this section were: number of iterations = 35,000, learning rate = 0.02, and *p* parameter = 0.9.

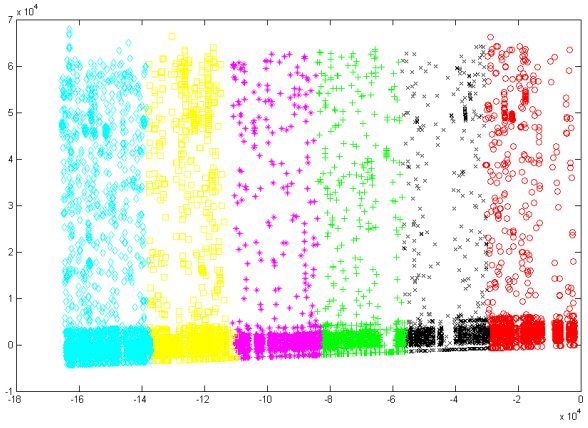

Fig. 31. PCA projection of 5-month data - Time.

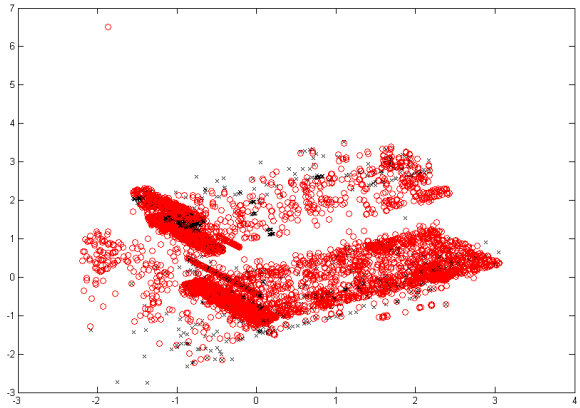

Fig. 32. MLHL projection of 5-month data - Snort output.

Fig. 32 shows the MLHL projection of the case study 2 by using the Snort output.

Fig. 33 shows the MLHL projection by considering the source port to depict the packets; from 3 to 10924: red circles, from 10925 to 21846: black crosses, from 21847 to 32768: green pluses, from 32769 to 43690: magenta stars, from 43691 to 54612: yellow squares, and from 54613 to 65529: cyan diamonds.

Those two visualizations show sharply the conclusions extracted before, according to Figs. 20 and 24.

5.2.2.3 Self-Organizing Map

For the SOM, the following options and parameters were tuned: grid size, batch/online training, initialization, number of iterations and distance criterion

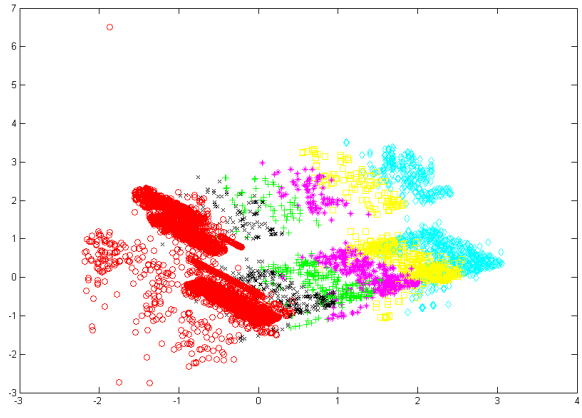

Fig. 33. MLHL projection of 5-month data - Source port.

among others. The used parameter values were: linear initialization, batch training, hexagonal lattice, "Gaussian" neighborhood function, and grid size (determined by means of a heuristic formula) = 22x29.
Fig. 34 shows the U-matrix of the SOM mapping of the case study 1.
Fig. 35 shows the SOM map of the case study 2 by using the protocol; 1 represents ICMP, 3 UDP and 5 TCP. It can be seen in this mapping that the SOM clusters the data in three big clusters. Each one of them contains all the traffic related with each one of the protocols, as occurred for the case of study one (ICMP: upper left corner, UDP: upper right corner, and TCP: the rest of the mapping).
Fig. 36 shows the SOM map of the case study 2 by using the Snort output; 1 represents the packets identified as attacks while 0 represents the undetected attacks. As pointed out for the case study 1, SOM is not able to cluster the data distinguishing the traffic classification of Snort (alarm/no alarm).

6 Conclusions and Future Work

Apart from the previously stated conclusions, regarding each one of the analyzed dataset, some other (more general) conclusions are gathered in this section.
After comparing the different projections obtained in this study, it can be concluded that CMLHL provides a more sparse and clearer representation than the other applied projection methods. This enables the intuitive visualization of the Honeynet data, where the general structure of these data can be seen and interpreted. The visualizations obtained through CMLHL give an insight

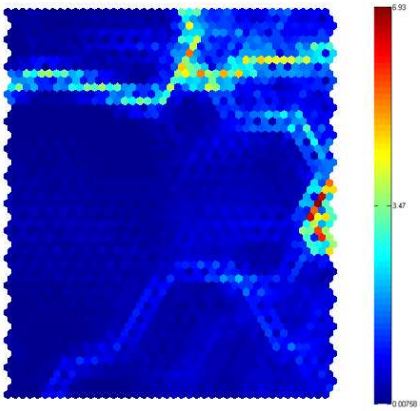

Fig. 34. SOM mapping of 5-month data - U-matrix.

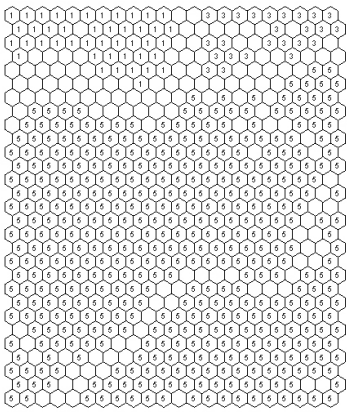

Fig. 35. SOM mapping of 5-month data - Protocol.

of the captured honeynet data, providing useful knowledge about the attacks a network could face.
It has been shown how CMLHL provides a helpful technique to identify backscatter attacks, as well as identifying those attacks that overflow a buffer and malware downloads.
From a general perspective, it can be seen from all the visualizations the high classification error rate of Snort. In keeping with this idea, it can be concluded that every IDS needs to be tuned and that default signatures are not enough to detect and identify every single attack.
After getting a general idea of the dataset structure, an in-deep analysis was carried out to comprehensively analyze each one of the points in the groups identified by CMLHL. As a result, the following conclusions can be stated for each one of the destination ports:

- 8: We identify the type of ICMP packet by inserting its code into the field destination port. ICMP type 8

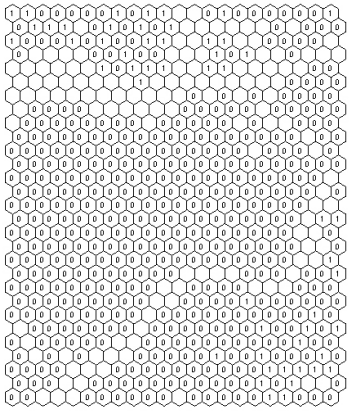

Fig. 36. SOM mapping of 5-month data - Snort output.

corresponds to ICMP echo or ping, used for probing the Internet, looking for victim hosts.

- 22: SSH. It seems to be a traffic flow with many packets coming from one source to one of the honeypot. They correspond to connection attempts by attackers or infected machines.
- 80: HTTP. Attackers try different vulnerabilities against web applications.
- 135: DCE endpoint resolution, used by Microsoft for Remote Procedure Call protocol. It has always been and still is one of the most exploited services by virus and worms.
- 139: NETBIOS Session Service. Plenty of attacks to this Microsoft Windows service can be found.
- 443: HTTP protocol over TLS SSL connection attempts.
- 445: SMB directly over IP. As most of the traffic in the biggest group identified by CMLHL is aimed at this destination port, we can conclude that this is a widely exploited service.
- 1433: Microsoft-SQL-Server, used by the old SQL Slammer worm.
- 1521: Oracle TNS Listener. It seems that attackers try to connect to the honeypot via Oracle service.
- 2967: Symantec System Center. Vulnerabilities have been found on Symantec service, and it is being expiated in the wild.
- 3128: Proxy Server // Reverse WWW Tunnel Backdoor, where the MyDoom worm operates.
- 3389: MS Terminal Services, used for Remote Desktop.
- 4444: This port is a common return port for the rpc dcom.c buffer overflow vulnerability and for the msblast rpc worm.
- 4899: Remote Administrator default port. There is a known remotely exploitable vulnerability in radmin server versions 2.0 and 2.1 that allows code execution.
- 5061: SIP-TLS. Used for VoIP communications.
- 5900: Virtual Network Computer or VNC, used also as a remote desktop solution.
- Port 8080: HTTP Alternate port, also used as an HTTP proxy.
- Port 19765: Used in Kademlia (Bittorrent protocol).

This deeper analysis remains necessary in order to better understand some of the visualized attacks, but CMLHL projections seem enough to obtain a fast understanding of Internet attacks.

Further work will focus on the application of different projection/visualization models as well as studying the visualization with different metrics instead of using the original features of the data.

More analysis can be done with the data, like visualization of this attack traffic by each of the honeynet infrastructure sensors. This way, one could compare the pattern of attack behavior distinguishing the Internet space placement.

Another interesting improvement of CMLHL visualization could be providing interactive capabilities; a user or analyst could select one or more points from the projections and the system may give details about the data behind them. In a further approach, the system could automatically generate signatures for user selected clusters, giving a solution to the big amount of Snort's undetected packets.

Enrichment of the attack dataset may also be a focus of attention. Researchers are correlating network traffic data with exploits collected during simulated vulnerability exploitation. Malware is also obtained for those attacks that aim to spread the infection. All this data needs a deeper analysis, and neural projection techniques will help in that task.

Acknowledgements

This research has been partially supported through the Regional Government of Castilla y León under Project BU006A08, the Regional Government of Gipuzkoa, the Department of Research, Education and Universities of the Basque Government; and the Spanish Ministry of Science and Innovation (MICINN) under project TIN2010-21272-C02-01 and CIT-020000-2009-12 (funded by the European Regional Development Fund). The authors would also like to thank the vehicle interior manufacturer, Grupo Antolin Ingenieria S.A., within the framework of the MAGNO2008 - 1028.- CENIT Project also funded by the MICINN.

Appendix A - Remaining Visualizations

6.1 Case study 1: a 1-month dataset

6.1.1.1 Principal Component Analysis

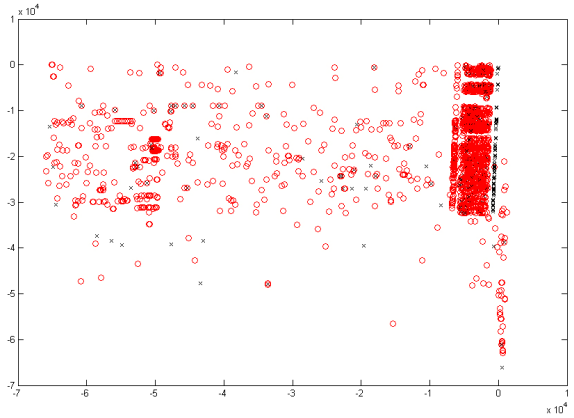

Fig. 37. PCA projection of 1-month data - Snort output.

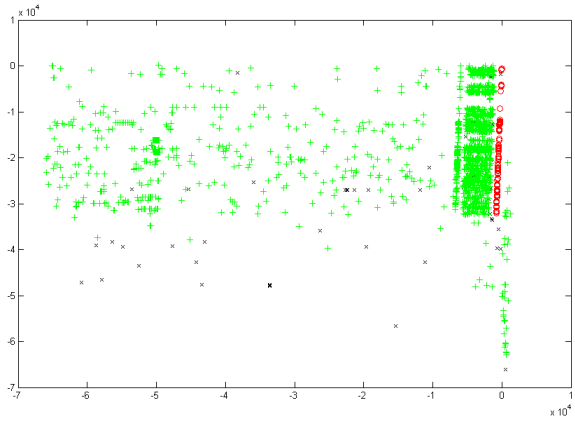

Fig. 38. PCA projection of 1-month data - Protocol.

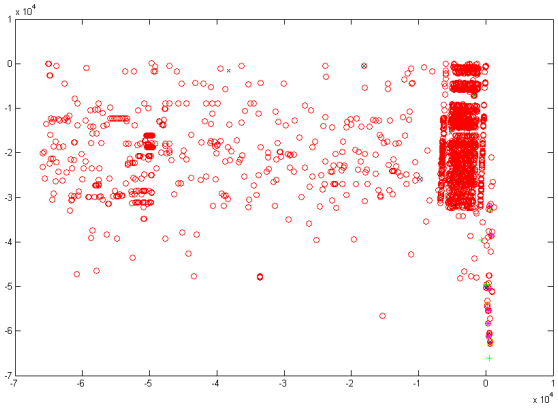

Fig. 39. PCA projection of 1-month data - IP Length.

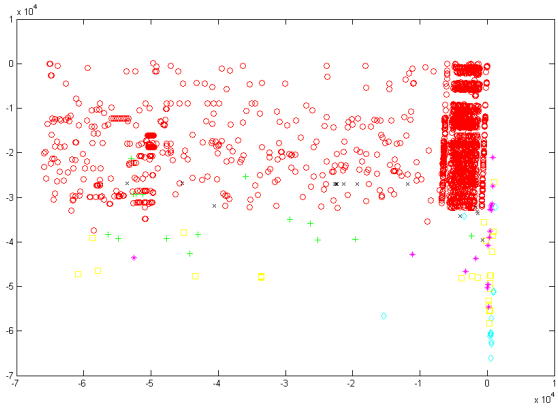

Fig. 40. PCA projection of 1-month data - Destination port.

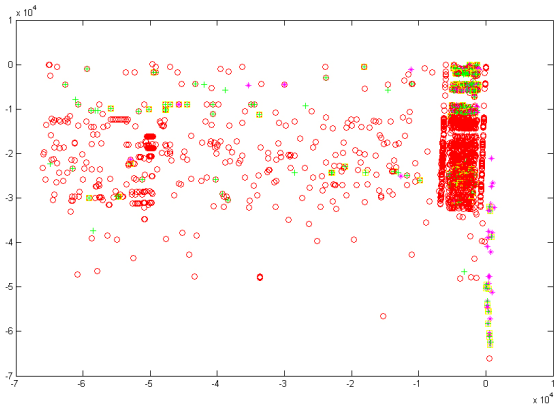

Fig. 41. PCA projection of 1-month data - Flags.

6.1.1.2 Maximum Likelihood Hebbian Learning

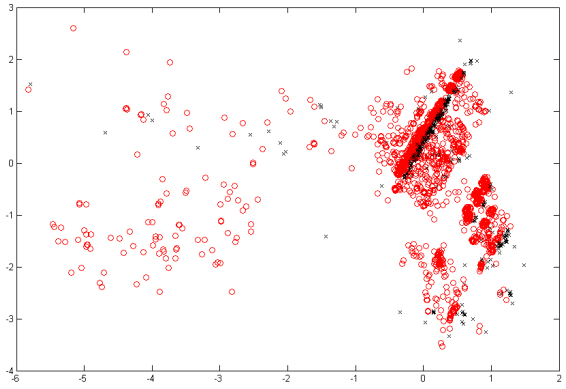

Fig. 42. MLHL projection of 1-month data - Snort output.

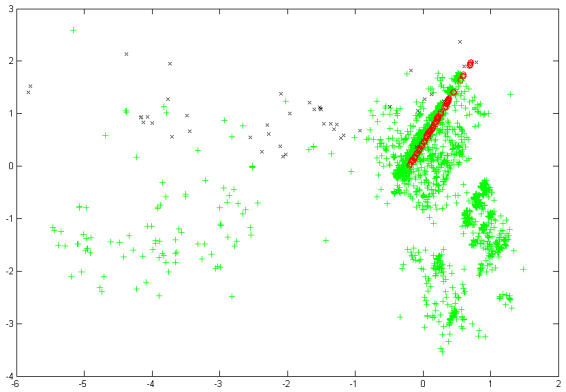

Fig. 43. MLHL projection of 1-month data - Protocol.

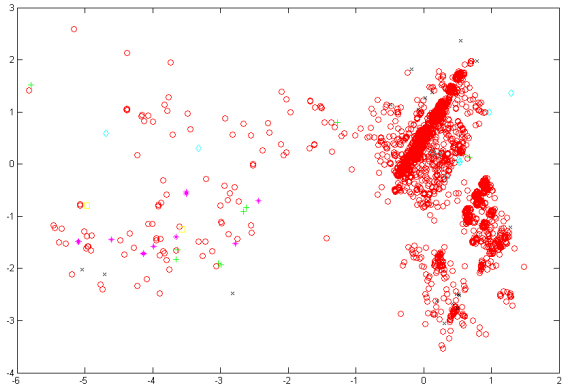

Fig. 44. MLHL projection of 1-month data - IP Length.

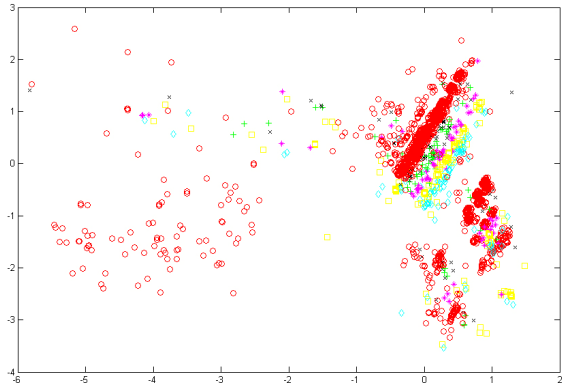

Fig. 45. MLHL projection of 1-month data - Source port.

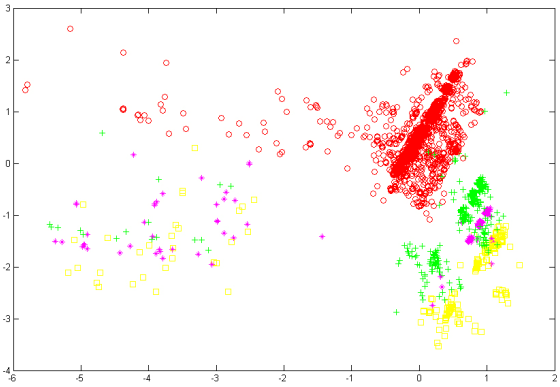

Fig. 46. MLHL projection of 1-month data - Flags.

6.1.1.3 Curvilinear Component Analysis

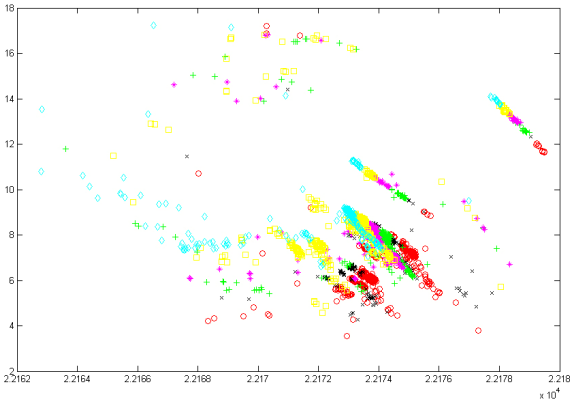

Fig. 47. CCA projection of 1-month data - Time.

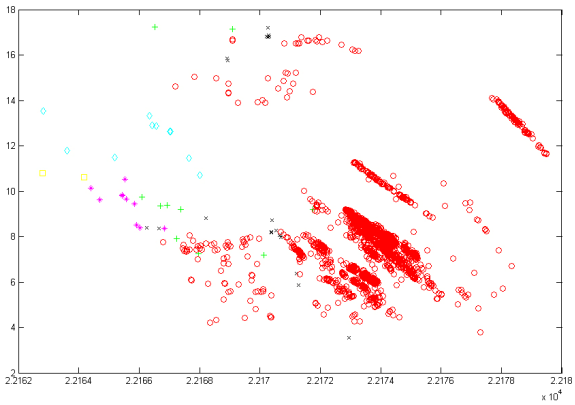

Fig. 48. CCA projection of 1-month data - IP length.

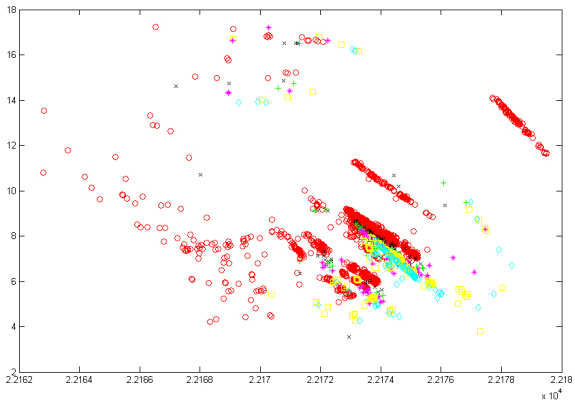

Fig. 49. CCA projection of 1-month data - Source port.

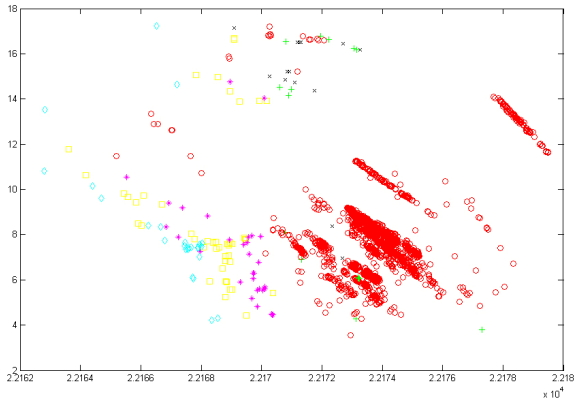

Fig. 50. CCA projection of 1-month data - Destination port.

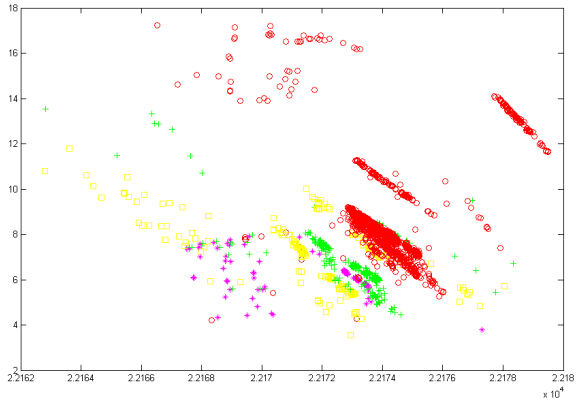

Fig. 51. CCA projection of 1-month data - Flags.

6.1.1.4 Self-Organizing Map

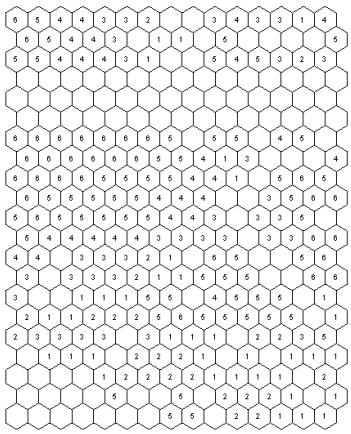

Fig. 52. SOM mapping of 1-month data - Time.

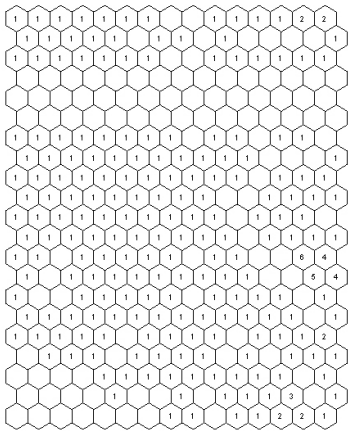

Fig. 53. SOM mapping of 1-month data - IP length.

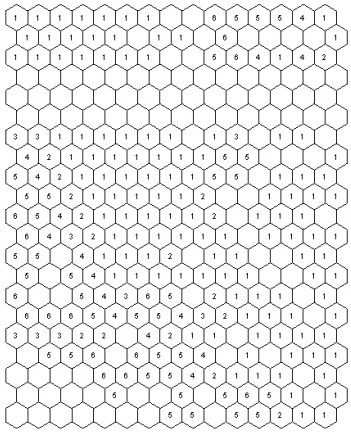

Fig. 54. CCA projection of 1-month data - Source port.

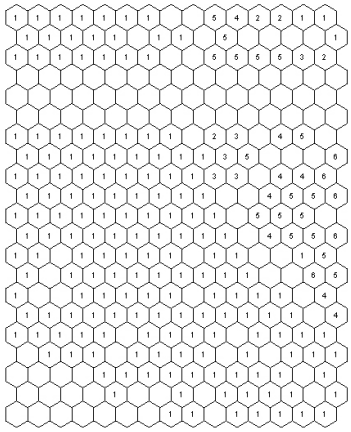

Fig. 55. CCA projection of 1-month data - Destination port.

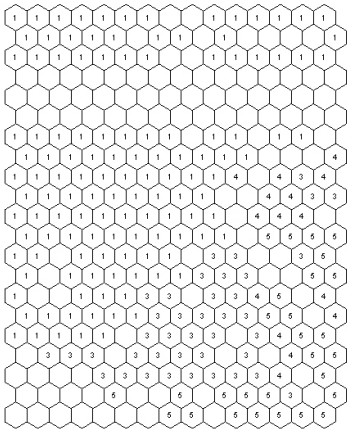

Fig. 56. CCA projection of 1-month data - Flags.

References

1. J. M. Myerson, Identifying enterprise network vulnerabilities, *International Journal of Network Management*, 12 (3) (2002) 135-144.
2. Computer security threat monitoring and surveillance, (James P. Anderson Co 1980).
3. D. E. Denning, An intrusion-detection model, *IEEE Transactions on Software Engineering*, 13 (2) (1987) 222-232.
4. T. Chih-Fong, H. Yu-Feng, L. Chia-Ying, and L. Wei-Yang, Intrusion detection by machine learning: A review, *Expert Systems with Applications*, 36 (10) (2009) 11994-12000.
5. R. A. Becker, S. G. Eick, and A. R. Wilks, Visualizing network data, *IEEE Transactions on Visualization and Computer Graphics*, 1 (1) (1995) 16-28.
6. A. D. D'Amico, J. R. Goodall, D. R. Tesone, and J. K. Kopylec, Visual discovery in computer network defense, *IEEE Computer Graphics and Applications*, 27 (5) (2007) 20-27.
7. J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi, Focusing on context in network traffic analysis, *IEEE Computer Graphics and Applications*, 26 (2) (2006) 72-80.
8. T. Itoh, H. Takakura, A. Sawada, and K. Koyamada, Hierarchical visualization of network intrusion detection data, *IEEE Computer Graphics and Applications*, 26 (2) (2006) 40-47.
9. Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher, and S. Foresti, A visualization paradigm for network intrusion detection, in *Sixth Annual IEEE SMC Information Assurance Workshop, 2005. IAW '05*,eds., 2005) pp. 92-99.
10. H. Adeli and S. L. Hung, *Machine learning - neural networks, genetic algorithms and fuzzy systems*, (John Wiley & Sons, New York, USA, 1995).
11. H. Adeli and X. Jiang, Dynamic fuzzy wavelet neural network model for structural system identification, *Journal of Structural Engineering*, 132 (2006) 102-111.
12. A. Mehran and A. Hojjat, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integrated Computer-Aided Engineering*, 17 (3) (2010) 197-210.
13. M. Oja, G. O. Sperber, J. Blomberg, and S. Kaski, Self-organizing map-based discovery and visualization of human endogenous retroviral sequence groups, *International Journal of Neural Systems*, 15 (3) (2005) 163-180.
14. Á. Herrero, E. Corchado, P. Gastaldo, and R. Zunino, Neural projection techniques for the visual inspection of network traffic, *Neurocomputing*, 72 (16-18) (2009) 3649-3658.
15. Á. Herrero, E. Corchado, M. A. Pellicer, and A. Abraham, Movih-ids: A mobile-visualization hybrid intrusion detection system, *Neurocomputing*, 72 (13-15) (2009) 2775-2784.
16. E. Corchado and Á. Herrero, Neural visualization of network traffic data for intrusion detection, *Applied Soft Computing*, ("Accepted - In press") (2010).
17. E. Corchado, D. MacDonald, and C. Fyfe, Maximum and minimum likelihood hebbian learning for exploratory projection pursuit, *Data Mining and Knowledge Discovery*, 8 (3) (2004) 203-225.
18. L. Xu, Best harmony, unified rpcl and automated model selection for unsupervised and supervised learning on gaussian mixtures, three-layer nets and me-rbf-svm models, *International Journal of Neural Systems*, 11 (1) (2001) 43-69.
19. K. A. Charles, Decoy systems: A new player in network security and computer incident response, *International Journal of Digital Evidence*, 2 (3) (2004).
20. N. Provos, A virtual honeypot framework, in *13th USENIX Security Symposium*,eds., 2004) pp.
21. P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, The nepenthes platform: An efficient approach to collect malware, in *9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*,eds. (Springer Berlin / Heidelberg, 2006) pp. 165-184.
22. D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, Inferring internet denial-of-service activity, *ACM Transactions on Computer Systems*, 24 (2) (2006) 115-139.
23. J. McHugh, A. Christie, and J. Allen, Defending yourself: The role of intrusion detection systems, *IEEE Software*, 17 (5) (2000) 42-51.
24. L. Khaled, Computer security and intrusion detection, *Crossroads*, 11 (1) (2004) 2-2.
25. J. P. Anderson, Computer security threat monitoring and surveillance, (Technical Report 1980).
26. P. G. Neumann and D. B. Parker, A summary of computer misuse techniques, in *12th National Computer Security Conference*,eds., 1989) pp. 396-407.
27. A. Sundaram, An introduction to intrusion detection, *Crossroads*, 2 (4) (1996) 3-7.
28. P. Laskov, P. Dussel, C. Schafer, and K. Rieck, Learning intrusion detection: Supervised or unsupervised?, in *13th International Conference on Image Analysis and Processing (ICIAP 2005)*,eds. F. Roli and S. Vitulano (Springer, Heidelberg, 2005) pp. 50-57.
29. J. M. Rizza, *Computer network security*, (Springer US, 2005).
30. I. Balepin, S. Maltsev, J. Rowe, and K. Levitt, Using specification-based intrusion detection for automated response, in *Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*,eds. (Springer, Heiderlberg, 2003) pp. 136-154.
31. Libpcap, http://www-nrg.ee.lbl.gov/. Last access:
32. S. M. Bellovin, There be dragons, in *Third Usenix Security Symposium*, Citeseer, 1992), pp. 14-16.

33. B. Cheswick, An evening with berferd in which a cracker is lured, endured, and studied,eds. (Citeseer, 1990) pp. 1990.
34. Deception toolkit, http://www.all.net/dtk/. Last access:
35. L. Spitzner, *Honeypots: Tracking hackers*, (Addison-Wesley Professional, 2003).
36. L. Spitzner, Honeypots: Sticking to hackers, *Network Magazine*, 18 (4) (2003) 48-51.
37. C. Ahlberg and B. Shneiderman, Visual information seeking: Tight coupling of dynamic query filters with starfield displays, in *Readings in information visualization: Using vision to think* Morgan Kaufmann Publishers Inc., 1999), pp. 244-250.
38. J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi, Preserving the big picture: Visual network traffic analysis with tnv, in *IEEE Workshop on Visualization for Computer Security (VizSEC 05)*,eds. (IEEE Computer Society, 2005) pp. 47-54.
39. J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, State of the practice of intrusion detection technologies, (Carnegie Mellon University - Software Engineering Institute 2000).
40. T.-W. Yue and S. Chiang, A neural-network approach for visual cryptography and authorization, *International Journal of Neural Systems*, 14 (3) (2004) 175-187.
41. J. Zhou, Q. Z. Xu, W. J. Pei, Z. He, and H. Szu, Step to improve neural cryptography against flipping attacks, *International Journal of Neural Systems*, 14 (6) (2004) 393-405.
42. T. Goldring, Scatter (and other) plots for visualizing user profiling data and network traffic, in *2004 ACM Workshop on Visualization and Data Mining for Computer Security*,eds. (ACM Press, 2004) pp. 119-123.
43. A. Lazarevic, V. Kumar, and J. Srivastava, Intrusion detection: A survey, in *Managing cyber threats: Issues, approaches, and challenges* Springer US, 2005), pp. 19-78.
44. K. Pearson, On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, 2 (6) (1901) 559-572.
45. H. Hotelling, Analysis of a complex of statistical variables into principal components, *Journal of Education Psychology*, 24 (1933) 417-444.
46. C. M. Bishop, *Neural networks for pattern recognition*, (Oxford University Press, 1996).
47. E. Oja, Neural networks, principal components, and subspaces, *International Journal of Neural Systems*, 1 (1989) 61-68.
48. E. Oja, Principal components, minor components, and linear neural networks, *Neural Networks*, 5 (6) (1992) 927-935.
49. E. Oja, A simplified neuron model as a principal component analyzer, *Journal of Mathematical Biology*, 15 (3) (1982) 267-273.
50. D. Sanger, Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks, *Connection Science*, 1 (2) (1989) 115-138.
51. C. Fyfe, A neural network for pca and beyond, *Neural Processing Letters*, 6 (1-2) (1997) 33-41.
52. E. Corchado and C. Fyfe, Connectionist techniques for the identification and suppression of interfering underlying factors, *International Journal of Pattern Recognition and Artificial Intelligence*, 17 (8) (2003) 1447-1466.
53. J. H. Friedman and J. W. Tukey, A projection pursuit algorithm for exploratory data-analysis, *IEEE Transactions on Computers*, 23 (9) (1974) 881-890.
54. E. Corchado, Y. Han, and C. Fyfe, Structuring global responses of local filters using lateral connections, *Journal of Experimental & Theoretical Artificial Intelligence*, 15 (4) (2003) 473-487.
55. H. S. Seung, N. D. Socci, and D. Lee, The rectified gaussian distribution, *Advances in Neural Information Processing Systems*, 10 (1998) 350-356.
56. P. Demartines and J. Herault, Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets, *IEEE Transactions on Neural Networks*, 8 (1) (1997) 148-154.
57. P. Demartines, Analyze de données par réseaux de neurones auto-organizés, Institut National Polytechnique de Grenoble, 1994.
58. T. Kohonen, The self-organizing map, *IEEE*, 78 (9) (1990) 1464-1480.
59. D. W. Patterson, *Artificial neural networks: Theory and applications*, (Prentice Hall PTR Upper Saddle River, NJ, USA, 1998).
60. T. Kohonen, *Self-organizing maps*, (Springer-Verlag New York, Inc., 1997).
61. B. Baruque and E. Corchado, A weighted voting summarization of som ensembles, *Data Mining and Knowledge Discovery*, (in press) (2010).
62. H. Ritter, T. Martinetz, and K. Schulten, *Neural computation and self-organizing maps; an introduction*, (Addison-Wesley Longman Publishing Co., Inc., 1992).
63. R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, The 1999 darpa off-line intrusion detection evaluation, *Computer Networks*, 34 (4) (2000) 579-595.
64. R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation, in *DARPA Information Survivability Conference and Exposition (DISCEX '00)*,eds., 2000) pp. 12-26.
65. R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, Analysis and results of the 1999 darpa off-line intrusion detection evaluation, in *Third International Workshop on the Recent Advances in Intrusion Detection (RAID 2000)*,eds. H. Debar, L. Mé, and S. F. Wu (Springer, Heidelberg, 2000) pp. 162-182.
66. Kdd cup 1999 dataset, http://archive.ics.uci.edu/ml/databases/kddcup99/kddcup99.html. Last access: 24/06/2009

67. M. Elkan, Results of the kdd'99 classifier learning contest, http://www-cse.ucsd.edu/users/elkan/clresults.html, 1999.
68. Euskalert, basque honeypot network, http://www.euskalert.net. Last access: 10/05/2010
69. U. Zurutuza, R. Uribeetxeberria, and D. Zamboni, A data mining approach for analysis of worm activity through automatic signature generation, in *1st ACM Workshop on AISec*,eds. (ACM, 2008) pp. 61-70.
70. P. Ruoming, Y. Vinod, B. Paul, P. Vern, and P. Larry, Characteristics of internet background radiation, in *4th ACM SIGCOMM Conference on Internet Measurement*,eds. (ACM, 2004) pp. 27-40.