Neurocomputing 72 (2009) 2754–2762

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Explanations of unsupervised learning clustering applied to data security analysis

G. Corral [a,*], E. Armengol [b], A. Fornells [a], E. Golobardes [a]

[a] Grup de Recerca en Sistemes Intelligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins, 2, 08022 Barcelona, Spain
[b] IIIA, Artificial Intelligence Research Institute, CSIC, Spanish Council for Scientific Research, Campus UAB, 08193 Bellaterra, Barcelona, Spain

ARTICLE INFO

Available online 16 April 2009

*Keywords:*
Unsupervised learning clustering
Explanations
Self-organizing maps
Network security
Artificial intelligence applications

ABSTRACT

Network security tests should be periodically conducted to detect vulnerabilities before they are exploited. However, analysis of testing results is resource intensive with many data and requires expertise because it is an unsupervised domain. This paper presents how to automate and improve this analysis through the identification and explanation of device groups with similar vulnerabilities. Clustering is used for discovering hidden patterns and abnormal behaviors. Self-organizing maps are preferred due to their soft computing capabilities. Explanations based on anti-unification give comprehensive descriptions of clustering results to analysts. This approach is integrated in *Consensus*, a computer-aided system to detect network vulnerabilities.



## 1. Introduction

Nowadays nobody conceives a competitive organization without all its resources interconnected and continuously available. As organizations have become increasingly dependent on their networks and Internet, they have also opened themselves up to potentially immense risks and vulnerabilities. Consequently periodically audits and vulnerability assessments are needed to detect and eliminate these possible security holes. Vulnerability assessment is the process of identifying and quantifying vulnerabilities in a system or a network [34]. A system is considered as a network device in this domain, that is to say a computer, application server, peripheral or other related communications equipment attached to a network. However, time and cost can limit the depth of a vulnerability assessment. These limitations justify the automation of the processes involved in a vulnerability assessment, especially those related to the analysis of test results, as a thorough network security test generates large quantities of data.

Comprehensive network security analysis must coordinate diverse sources of information to support large scale visualization and intelligent response [34]. Thus it is prohibitively expensive to classify them manually. Security applications require of some intelligence to recognize malicious data, unauthorized traffic, intrusion data patterns, learn from previous decisions and also provide a proactive security policy implementation [17,18]. Artificial intelligence (AI) techniques can help managing all this information. Unsupervised learning is suitable to handle vulnerability assessment results. Security applications do not provide data with labelled classes or any other information beyond the raw data. Then unsupervised learning can help discovering subjacent patterns of behavior between tested devices. Also soft computing techniques are adequate to this environment, as their properties are useful when dealing with imprecision, uncertainty and partial truth. Data compiled from different sources of information can lead to incongruent or imprecise information, data eligible for soft computing. On the other hand, security analysts expect a brief account that explains AI results in an understandable language for them.

This paper presents a proposal in order to cope with all these issues. The process of gathering data from network security tests has been already automated with a system called *Consensus* [11]. This computer-aided system stores data obtained from network vulnerability assessments to show it to security analysts [11]. *Consensus* also integrates an analysis module for all the collected data, called *Analia*. Unsupervised learning is included in *Analia* with the aim of clustering vulnerability assessment results [12,13] and a simple explanation of clustering results is shown to analysts [12]. Security analysts obtain a solution with several clusters, and each cluster contains tested devices with similar vulnerabilities and behavior. When identifying and solving the vulnerabilities of a network device, the same process can be applied to all devices in that cluster. Also main efforts can be focused first on the most critical clusters without having to analyze the whole data set. This paper presents an improvement of *Analia*, where different

* Corresponding author.
*E-mail addresses:* guiomar@salle.url.edu (G. Corral), eva@iiia.csic.es (E. Armengol), afornells@salle.url.edu (A. Fornells), elisabet@salle.url.edu (E. Golobardes).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.09.021

clustering techniques have been tested in this unsupervised domain, and more understandable explanations are given to security analysts, independently of the selected clustering technique. These enhanced explanations will be based on the Anti-unification (AU) algorithm [2] and will characterize each obtained cluster. These explications allow describing a cluster with the same representation language used to describe the elements; therefore security analysts can understand results more easily. Analysts will obtain a solution where all tested devices have been grouped in different clusters regarding their vulnerabilities and will know the reason of these clustering results. Moreover, these explanations will be useful to detect not only common features between the elements of a cluster, but also misconfigurations or rogue devices.

This paper is organized as follows: Section 2 surveys related work about AI applied in network security and the use of explanations. Section 3 describes *Consensus* and mainly, its analysis module, *Analia*. Section 4 details the clustering solution. Section 5 explains the proposal of explanations. Section 6 describes the experiments. Finally, Section 7 presents conclusions and further work.

## 2. Related and previous work

A network security test generates large amount of data. Several testing tools are needed to audit network devices. As a result, the analysis of this data must coordinate diverse sources of information. These inputs may not always agree in the same results, providing complex data with imprecision and uncertainty. A manual classification of this information becomes an exhausting labour [28]. Thus AI, and more specifically soft computing techniques, can be used to build security systems that exploit a tolerance for imprecision and uncertainty. These new systems will provide a more tractable, robust, low-cost solution and will closely resemble the human decision-making process [10].

The analysis of information provided by network security tests requires improved methods to identify behavior patterns, malicious data or unauthorized changes in a network [17]. Unsupervised learning can be applied in this security test environment, where the domain and the different classes have not been defined and no previous knowledge of network behavior and data results is required. In this sense, unsupervised learning can be used to cluster groups of tested devices with similar vulnerabilities and find features inherent to the problem.

A large number of clustering methods exist. The choice of a clustering algorithm depends on the type of available data and on the particular purpose [22]. Then algorithms closer to our investigation will be enumerated. Focusing on the inductive approach, partition methods cluster training data into $K$ clusters, where $K<M$ and $M$ is the number of objects in the data set. One of the most representative partition methods is the $K$-means algorithm [24]. Its main drawback is the configuration of the $K$ parameter. In this sense, the $X$-means [35] algorithm automatically calculates the best number of clusters for each execution. Model-based methods use mathematical and probabilistic models; an example is the Autoclass algorithm [9]. Grid-based methods divide the space into a finite number of cells that form a grid structure on which all clustering operations are performed. Density-based methods discover clusters with an arbitrary shape. Focusing on artificial neural networks (NN) [5], Kohonen map or self-organizing map (SOM) [27] is a widely used unsupervised learning model. Adaptive resonance theory (ART) is another NN technique and many different variations of ART are available. The simplest ART network classifies an input vector into a category depending on the stored pattern it most closely resembles [7].

Different clustering techniques have been applied in the network security domain. $K$-means has been used to find natural grouping of similar alarm records [6,21] and also to detect network intrusions [28,39]. SOM has already been applied as clustering method to study computer attacks [17], to detect network intrusions [18], and also to detect anomalous traffic [36].

Soft computing techniques can improve the discovery of implicit and previously unknown knowledge using a better feature extraction in a high dimensional space to cluster data [22]. An example of a soft computing technique is SOM [27].

However, a drawback of data clustering methods is their lack of interpretation of clustering results, as they do not explain why some elements are in the same cluster and other elements have been grouped separately. Hence a further interpretation is needed to facilitate an accurate explanation of the results.

The idea of explaining results comes from initial expert systems where explanations consisted on the chain of rules used to reach the final result (e.g., MYCIN [15]). This kind of explanations is specific for rule-based systems and then it is necessary to define some other kind of representations depending on the representation used by the system. Thus, case-based reasoning (CBR) systems [1] commonly use as explanation the set of cases that are the most similar to a given problem. Explanation-based learning methods [16] give as explanation of a result the generalization of the reasoning followed to prove that an example belongs to a class. Conceptual clustering assembles entities into a single cluster not because of their pairwise similarity, but because together they represent a concept from a predefined set of concepts [31]. Consequently, this approach not only clusters entities, but also provides concept descriptions of the obtained clusters [31]. However *a priori* conceptual entities must be defined [38] in conceptual clustering. This requirement is not always possible in several domains, like the network security domain, where analysts do not have a complete knowledge about the problem domain and can only deal with data instances themselves. A different method should be applied in this environment in order to give an explanation of clustering results to security analysts.

Different viewpoint of explanations can be found in the literature. On one hand, several authors have oriented their research to analyze how explanations have to change with either the usage of the system [4,8], or the user's expertise [30]. On the other hand, authors have focused their research on interacting with the user to refine the proposed solution, proposal commonly followed by recommender systems [23,29]. In particular, authors analyze which aspects are relevant to explain the result. Concerning CBR systems, the common explanation of showing the user the set of more similar cases seems not be the best one when cases have a complicated structure [19,30]. For that reason, an explanation should make explicit the contribution of each attribute to the classification problem [32], and even both similarities and differences are useful to explain the result [30]. In this context, an explanation scheme based on similarities for classification problems in CBR was proposed [3] using the anti-unification technique [2].

How the explanation scheme proposed in [3] could be adapted to explain the results of clustering methods? The main difference among methods applied to classification problems and clustering methods is that in latter methods the solution classes are not known. This means that no information related to common features in a class can be used, as classes have not been previously defined. As a result, the proposal described in next sections presents an explanation scheme that shows the features that all the elements of a cluster have in common to justify their membership to the same cluster. This proposal is applied to the

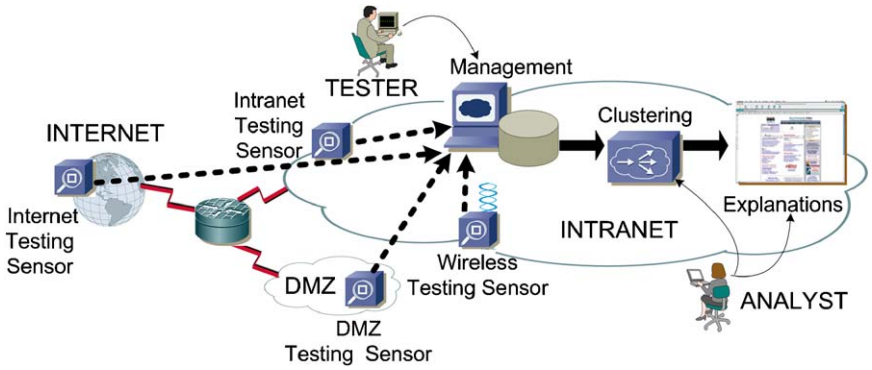


**Fig. 1.** *Consensus* data acquisition system and *Analia* data analysis module.

dataset of network security audits obtained from *Consensus* system [11] after clustering the dataset.

## 3. *Consensus* system and *Analia* analysis module

A vulnerability assessment pursues two main goals: test everything possible and generate a concise report. Consequently, two different profiles are needed: a network security tester and a network security analyst. They will perform different tasks.

*Analia* is the analysis module of *Consensus* that includes unsupervised learning to improve the analysis of information compiled after a network security test. *Consensus* automates the security testing procedures to reliably verify network security [11]. *Analia* uses data stored in *Consensus* to help security analysts finding hidden patterns in tested devices. The whole system is shown in Fig. 1. Different testing sensors of *Consensus* collect information valuable for a network vulnerability assessment and store this data in the management module. Security testers can control this automated process through the management console. After performing a security test, *Analia* is the module that analysts will use to visualize testing results.

After a security test, analysts must focus on data from every network device to find abnormal behaviors, incorrect configurations or critical vulnerabilities. If the list of devices is extensive, certain behaviors could become unnoticeable, some patterns could be masked or maybe the most vulnerable devices could be the last to be checked. *Analia* aids analysts to find similarities within data resulting from security tests using unsupervised learning. *Analia* helps analysts extracting conclusions without analyzing the whole data set. Finally, explanations are needed to justify clustering results.

This modular architecture offers different advantages. First, there is a separation between data collection and analysis. Network tests can be regularly planned and data is stored after every test. Whenever necessary, an analysis of the tested devices can be performed. *Analia* is a complementary module to refine results. *Analia* works with the same database where *Consensus* stores all data, avoiding data duplication. Its knowledge representation flexibility allows configuring different representations for the same data set as inputs to the clustering algorithms. Also the incorporation of new unsupervised learning techniques can be easily performed. Configuration files are obtained from the parameters selected by the analyst in the *Analia* web interface.

## 4. Clustering in Analia

The main goal of clustering is to group elements with similar attribute values into the same class. In the clustering task, classes are initially unknown and they need to be discovered from data. The clustering process usually involves the following steps: pattern representation and proximity measure definition, application of a clustering algorithm, data abstraction when needed and finally, validation of clustering results. This paper describes how clustering and data abstraction have been improved in *Analia*. The other phases have been already described in previous works [12,13].

Algorithms from different unsupervised learning approaches are included in *Analia* to analyze their behavior when applied to the security data set. First of all, a partition method like *K*-means [24] has been integrated in *Analia*. It conforms to the type of attributes of this problem and has been adapted to support missing values of attributes. However, this algorithm needs the final number of clusters ($K$) as input parameter and this is an unknown value. This unsupervised domain does not have previously defined how many different groups of tested devices will be found in a network. Then, several executions with different $K$ values need to be performed to choose the best one. On the other hand, *X*-means [35] automatically calculates the best value for $K$. In this sense, *X*-means has also been included in *Analia*.

Regarding to model-based methods, Autoclass [9] has been incorporated too. This technique can be configured to calculate the number of clusters automatically. The classes are described probabilistically, so that an object can have partial membership in the different classes and class definitions can overlap.

With respect to neural networks, SOM [27] has been selected to be included in *Analia* due to its soft computing capabilities [25], which allow SOM to work well in domains with a high number of dimensions with respect to the number of instances, manage uncertain knowledge, and provide an easy and intuitive way of showing groups of data. Previous experiments have demonstrated that SOM generates better results than the other implemented algorithms, not only qualitatively but also quantitatively [13]. Different validity indexes have been calculated and all of them indicate better clustering results when applying SOM. Also security analysts have been asked about the worthiness of the clustering results [13]. For this reason, next section and experimentation will primarily focus on experimentations using SOM as clustering technique.

### 4.1. Clustering in Analia using SOM

SOM [27] is one of the most well-known and used techniques for visualization and clustering data [26,33]. Because SOM is an unsupervised technique it has to discover by itself commonalities and correlations of the objects. Basically, it converts complex and nonlinear statistical relationships between high-dimensional data into simple geometric relationships on a low-dimensional space.

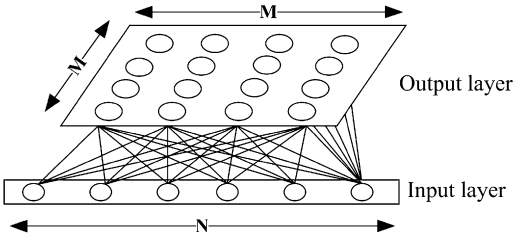


**Fig. 2.** SOM projects the original space from an input layer of $N$ neurons to an output layer of a new space with fewer dimensions in order to identify groups of similar elements.

In other words, it is able to identify groups of similar data according to their properties through a process which maintains the relations between data in the new low-dimensional space. Moreover, SOM is a soft computing technique that allows the management of uncertain, approximate, partial truth and complex knowledge [10]. These capabilities are appropriate for *Analia* domain.

Contrary to other neural networks approaches [5], the SOM's architecture is only composed of two layers. The input layer represents the new problem using as many neurons as input data features. On the other hand, the output layer describes the new low-dimensional space using as many neurons as the maximum number of expected clusters. Although the output layer can represent any dimensionality lower than the original $N$-dimensional space, it usually represents a grid of two dimensions such as the example of Fig. 2. In this case, the output layer contains $M \times M$ clusters, where each one is represented by a director vector of $N$ dimensions ($v_m$). A director vector can be described as the expected value for each one of the $N$ features. Finally, each input neuron is connected to all the output neurons.

Although there are many variants of the SOM training, the procedure can be detailed in the next steps:

1. Training examples are normalized between [0…1].
2. All the components of the director vectors are randomly initialized between [0…1].
3. Given a new training example $e$, the most suitable cluster is computed as the minimal distance between $e$ and each one of the director vectors. For example, the winner neuron can be the one with the lowest difference computed by the normalized Euclidean distance (see Eq. (1)).

$$similarity(e,m) = |1 - d(\overline{e}, \overline{v_m})| = \left| 1 - \sqrt{\frac{\sum_{n:1..N}(e(n) - v_m(n))^2}{N}} \right| \quad (1)$$

4. The director vector of the winner neuron is tuned to improve the match with new objects similar to the current one. In contrast, the director vectors of the neighbor neurons are modified to weakly represent the current example. The neighbor neurons are identified through the neighbor factor. On the other hand, the degree of adjustment is described by the learning factor. Both factors change through the training process. Initially their influence changes a lot the network while their effect is minimal at the end.
5. Steps 3 and 4 are repeated for all training examples until all the director vectors are representative enough. Usually their representation is determined by establishing a minimal error value. This value is computed as the global sum of the distance between the set of data of each cluster and its respective director vector. Nevertheless, another common criterion is to establish a maximum number of algorithm iterations.

When the training process ends, step 3 is the procedure used to map the new input example in the most suitable clusters.

To conclude, SOM is a smart technique to identify hidden and complex relationships between elements and also to identify the most relevant features thanks to its knowledge discovery and soft computing capabilities. Its benefits can be summarized in the following aspects [25]: it preserves the original topology; it works well even though the original space has a high number of dimensions; it incorporates the selection feature approach; although one class has few examples they are not lost; it provides an easy way to show data; it is organized in an autonomous way to be better adjusted to uncertain and complex data. This is exactly what experts need: the discovery of relationships between elements from network security audits to obtain groups of tested devices with similar security vulnerabilities. On the other hand, the main drawback is that SOM is influenced by the order of the training samples. In addition, it requires several training decisions which are not trivial to solve.

After a SOM execution, security analysts obtain a clustering configuration where each cluster contains network devices with similar vulnerabilities. Analysts can study each cluster separately in order to apply different security policies. However, they do not obtain any justification about that clustering result. SOM represents a cluster by its director vector, but this is not understandable information for analysts. A summary description of each cluster would be easier to comprehend and intuitively appealing for security analysts, especially when using the same vocabulary of feature characterization. This issue will be solved in the next section.

## 5. Explanation of clustering results in Analia

Classifications obtained by data clustering methods are often difficult to interpret conceptually. In most cases, they require an additional interpretation since no conceptual description is provided. This point has been already corroborated in *Analia* [12], due to the fact that results provided by algorithms like *K-means* [24], *X-means* [35], Auto-class [9] and SOM [27] group devices according to data attributes but do not explain the reason of that grouping. Afterwards a further interpretation is needed to give an accurate explanation of the clustering results to network security experts. This section presents a new approach to explain clustering results adapting the Anti-unification algorithm.

### 5.1. The Anti-unification algorithm in Analia

Previous work has enriched the *Analia* module by creating generalizations based on the anti-unification concept [2] to characterize each cluster [12]. These explanations depict a cluster with the same language used to represent the elements; therefore security analysts can easily interpret results. The main goal of anti-unification is to obtain a new element that contains all the common features of the different elements of a cluster. Any feature not shared between all the elements of a cluster will not be added to this new element. This new element will represent the whole cluster to explain why these devices have been grouped together.

Let $C$ be a cluster and let $e_1,\ldots,e_n$ be the set of elements of that cluster after the application of a clustering algorithm included in *Analia*. Each element $e_j$ is described by a set of attributes $A$, where each attribute $a_k \in A$ takes values in $V_k$. The Anti-unification

```
E: set of elements of a cluster
A: set of attributes describing the elements of E
Vk: set of all the possible values of the attribute ak
D: antiunification of the elements of E

function AU
   for each attribute ak in A
      Step 1 - if ak = v for all ei in E, then ak takes value v in D

      Step 2 - Let vki the value that each ei∈E holds in ak.
               if union(vki) = Vk  then ak will not appear in D
               if a vki = null value then ak will not appear in D

      Step 3 - otherwise ak = union(vki)

   end for
end-function AU
```

**Fig. 3.** Main steps of the AU algorithm adapted to *Analia* domain.

**Table 1**
Description of four elements from *Analia* dataset and their explanation *D* obtained from AU algorithm.

| | Ports | | | | | | W2000 | | XP | W2003 | Security notes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 21 | 25 | 53 | 80 | 135 | 445 | SP3 (%) | SP4 (%) | SP2 (%) | Server (%) | |
| $e_1$ | 1 | – | – | – | 1 | 1 | 75 | 75 | 75 | 75 | 3 |
| $e_2$ | 1 | 1 | 1 | – | 1 | 1 | 41 | 41 | 41 | 75 | 6 |
| $e_3$ | 1 | 1 | – | 1 | 1 | 1 | 41 | 41 | 41 | 75 | 6 |
| $e_4$ | 1 | 1 | – | – | 1 | 1 | 75 | 75 | 75 | 75 | 4 |
| *D* | 21:1 | – | – | – | 135:1 | 445:1 | W2000 SP3 | W2000 SP4 | XP SP2 | W2003 Server: 75% | Security notes |

algorithm builds a new feature term *D* to describe the cluster *C*. The main steps of AU adapted to *Analia* domain are shown in Fig. 3.

Finally, the anti-unification of the elements of a cluster *C* is a description *D* that contains the common attributes to all the elements in *C*. Those attributes with unknown value in any element $e_j \in C$ or those attributes that do not appear in some element $e_j \in C$ are not considered in constructing *D*. An attribute will not be either included in *D* when the union of the values that the attribute takes in the elements is the set of all possible values.

An example of AU adaptation over *Analia* dataset is shown in Table 1. Let *C* be a cluster formed by four elements, where $e_1$, $e_2$, $e_3$ and $e_4$ are network devices whose attributes detail their open ports, possible operating systems (OS) and the number of detected security notes. Attributes related to ports 25, 53 and 80 are not included in *D* due to the fact that some elements do not have any value assigned to them. Table 1 shows that attributes related to ports 21, 135 and 445 are present on all elements and their value is always '1'. This value means that these ports are all open in all the elements of the cluster. As these attributes contain the same value, this value is also reflected in *D*. Regarding to OS, only the attribute *W2003 Server* has the same value in all elements and hence this feature will have its value included in *D*. The rest of the attributes appear in all elements, but with different values; thus those attributes will be included in *D* with the union of those values. Finally, the element *D* reflects a device with ports 21, 135 and 445 open, which means that has FTP, MSRPC[1] and Microsoft-DS[2] services available; a probability of 75% of being a Windows 2003 Server, or maybe a Windows 2000 or an XP; and finally that an indeterminate number of security notes have been described.

[1] MSRPC: Microsoft Remote Procedure Call.
[2] Microsoft-DS: Port used for file sharing in Microsoft Windows.

The new element *D* can be used to detect regularities between all the elements of a given cluster. These regularities can help security analysts to give them a reason of that clustering result and to identify possible common problems present in all the elements of that cluster without having to study each element one by one.

However sometimes relevant information will never show up due to the inherent behavior of the AU algorithm. This technique only takes into account the features common to all elements. But in certain cases it should be useful to have information not only about common features to all elements, but also about common features to most of the elements of the cluster. In the example shown in Table 1, security analysts have corroborated that it would be profitable to include in *D* that the port 25, related to the protocol SMTP, was also open in most of the elements of cluster *C*. For that reason, the AU algorithm has been modified in order to process *Analia* dataset and give explanations of each cluster. These new explanations include the features common to all elements of a cluster and also incorporate the percentage of occurrence of the attributes not present in all elements.

### 5.2. The Detailed Anti-unification algorithm in Analia

The adaptation of the AU algorithm presented in this paper is shown in Fig. 4. This new algorithm is called Detailed_AU (DAU). In this adaptation, the description of a cluster will include the value of the attributes common to all the elements in that cluster and common only to some of them, showing the percentage of appearance of those values. An attribute will not be included in the description element when the union of the values that the attribute takes in the elements is the set of all possible values. After the execution, security analysts will be able to choose the relevance percentage to analyze.

```
E: set of elements of a cluster
A: set of attributes describing the elements of E
V_k: set of all the possible values of the attribute a_k
W_ki: percentage of elements with the attribute a_k holding the value v_i
D: anti-unification of the elements of E
W_k: set of all percentages of occurrence of values ∈ V_k of a_k ∈ A

function DAU

   for each attribute a_k in A

      step 1 - if a_k = v for all e_i in E, then
                   Store_in_D (a_k, v, W_k= 100%)

      step 2 - Let v_ki the value that each e_i∈E holds in a_k.
               if union(v_ki) = V_k  then a_k will not appear in D

      step 3 - otherwise
               Let V'_k = union(v_ki) (union of all the values that all e_i
               in E hold in a_k)
               Let n the number of elements in E
               for each v'_j in V'_k and v'_j ≠ null do
                   W_kj := 100 × (W_kj + 1) / n
                   Store_in_D (a_k, v'_j, W_kj)
               end for
   end for
end-function DAU
```

**Fig. 4.** The DAU algorithm constructs the explanation of a given set of elements in *D*.

**Table 2**
Description of four elements from *Analia* dataset and their explanation $D_i$ obtained from DAU algorithm.

| | Ports | | | | | | W2000 | | XP | W2003 | Security notes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 21 | 25 | 53 | 80 | 135 | 445 | SP3 (%) | SP4 (%) | SP2 (%) | Server (%) | |
| $e_1$ | 1 | – | – | – | 1 | 1 | 75 | 75 | 75 | 75 | 3 |
| $e_2$ | 1 | 1 | 1 | – | 1 | 1 | 41 | 41 | 41 | 75 | 6 |
| $e_3$ | 1 | 1 | – | 1 | 1 | 1 | 41 | 41 | 41 | 75 | 6 |
| $e_4$ | 1 | 1 | – | – | 1 | 1 | 75 | 75 | 75 | 75 | 4 |
| $D_i$ | 21:1<br>100% | 25:1<br>75% | 53:1<br>25% | 80:1<br>25% | 135:1<br>100% | 445:1<br>100% | W2000 SP3<br>41–50%<br>75–50% | W2000 SP4<br>41–50%<br>75–50% | XP SP2<br>41–50%<br>75–50% | W2003 Server<br>75–100% | Security notes<br>3–25%<br>4–25%<br>6–50% |

The DAU algorithm is independent of the clustering algorithm used to group elements. Once a clustering technique has been applied to the data set and elements have been collocated into different clusters, DAU is performed over each cluster to obtain an explanation of the elements gathered together.

An example of DAU application over *Analia* dataset is shown in Table 2. Let $C_i$ be a cluster formed by four elements, where $e_1$, $e_2$, $e_3$ and $e_4$ are network devices whose attributes detail their open ports, possible OS and the number of detected security notes. The explanation element $D_i$ contains the sharing of common attributes.

The summary explanation presented to a security analyst is shown in Fig. 5. This explanation is built using networking and security vocabulary useful for analysts. In case security analysts want to detect only attributes that are 100% common to all elements, DAU results would be equivalent to AU results. However, DAU allows detecting attributes common to most of

Description of cluster $C_i$:
It contains 4 elements.
Information of ports
- 100% of devices have port 21 open (FTP service open)
- 100% of devices have port 135 open (MSRPC service open)
- 100% of devices have port 445 open (Microsoft-DS service open)
- 75% of devices have port 25 open (SMTP service open)

Information of Operating Systems
- 100% of devices: Windows 2003 Server in a probability of 75%

Information of Security notes
- Different number of security notes

**Fig. 5.** Explanation of the elements of Table 2 obtained applying DAU algorithm, when the percentage of occurrence of attributes is selected to more than 70%.

the elements of a cluster, information that could be significant to analysts too. For example, in Table 2 there is a feature common to most of the elements: port 25, related to the SMTP protocol, is open in the 75% of the devices of that cluster. This feature would have become unnoticeable when applying AU, just because only one element of a cluster does not have this attribute defined, whereas all the other elements of the same cluster share this attribute. Consequently, DAU improves the information given to security analysts as attributes common to the majority of the elements can be also detected.

## 6. Experiments

The main goal of the experimentation is to corroborate that not only clustering but also explanations let analysts obtain groups of devices with similar vulnerabilities and understand each cluster characterization. *Analia* can also help detecting unauthorized changes. When testing similar devices, clustering should group them. If an element is in another cluster, descriptions of each cluster may explain the changes between the modified element and the rest. A comparison of their clustering descriptions will aid analysts detecting differences between elements that analysts had previously supposed that they should be in the same cluster sharing the same features.

Data sets have been extracted from real security tests performed over *La Salle-Universitat Ramon Llull* university network. These audits have been executed over public and internal servers, alumni laboratories and staff computers. The studied data set is composed of information obtained from 44 different devices: 21 (14+7) of two different labs, nine public servers, 11 internal servers and three staff computers. The analyzed data contains attributes related to all detected open ports, operating systems and vulnerability information of the different tested devices. All these attributes have been stored in *Consensus* database after the security audit over the university network.

The most restricted devices are alumni lab computers. They are administrated by the IT department and should follow the same pattern as any other software installation apart from a known list of applications is forbidden. Thus tested lab devices should be combined in the same cluster. If new software has been illegally installed in a device or its configuration has changed, it will be easy to identify when clustering, as this rogue device should be separated in a new cluster. The explanation of this new cluster should help analysts identify what has changed in that device.

In order to previously cluster the *Analia* dataset, SOM has been selected as a clustering technique due to its soft computing capabilities, as explained in Section 4. We have performed several SOM configurations with the purpose of finding out interesting results. SOM has been tested using several map sizes of 2-dimensions (from four clusters—$2 \times 2$—to 36 clusters—$6 \times 6$—to analyze several data dispersions), two different distance measures (normalized Euclidean and normalized Hamming distance) and 10 random seeds (to minimize the random effects of initialization). The learning factor, the neighbor factor and maximum iterations have been set to typical values: from 0.6 to 0.01, from $M$ to 1 and 500 iterations by neuron. Best executions have been selected using Dunn [20], DB [14], Silhouette [37] and Cohesion factors [13] as validity indexes.

Clustering results group devices with similar operating systems, open ports and detected vulnerabilities in the same cluster. Regarding to devices evaluated in the two alumni labs, elements should be grouped in two separated clusters as every lab has a different computer configuration. However, clustering results divide those elements in three clusters. One cluster contains 14 devices of the same lab, as expected. On the other hand, the seven devices of the second lab have been placed in two clusters, where one cluster contains six devices and another cluster contains only one device. These results are shown in Table 3a, where lab devices of the second laboratory are represented by their IP addresses in their respective clusters.

**Table 3**
(a) Clustering results with IP addresses of devices. (b) Clustering distribution of devices from network X = 10.0.14.0/24.

| Cluster 1 | Cluster 2 |
|---|---|
| (a) | |
| 10.0.14.206 | 10.0.14.203 |
| 10.0.14.207 | |
| 10.0.14.208 | |
| 10.0.14.209 | |
| 10.0.14.201 | |
| 10.0.14.202 | |

(b)

| | X.206 | X.207 | X.208 | X.209 | X.201 | X.202 | X.203 |
|---|---|---|---|---|---|---|---|
| X.206 | – | 100 | 100 | 100 | 100 | 85.7 | 0 |
| X.207 | 100 | – | 100 | 100 | 100 | 85.7 | 0 |
| X.208 | 100 | 100 | – | 100 | 100 | 85.7 | 0 |
| X.209 | 100 | 100 | 100 | – | 100 | 85.7 | 0 |
| X.201 | 100 | 100 | 100 | 100 | – | 85.7 | 0 |
| X.202 | 100 | 100 | 100 | 100 | 85.7 | – | 0 |
| X.203 | 0 | 0 | 0 | 0 | 0 | 0 | – |

Probability of devices of being clustered together.

Table 3b shows, for all SOM executions, how frequently a device of a row has been clustered altogether with a device of a column. This table shows that devices with IP addresses 10.0.14.201, 10.0.14.206, 10.0.14.207, 10.0.14.208 and 10.0.14.209 have been clustered always together, in the 100% of the executions; device 10.0.14.202 has been clustered with the former devices in the 85% of the executions, whereas device 10.0.14.203 has been never clustered with any other device for all selected executions. Other cluster results have been omitted for security purposes as they have public IP addresses. Security analysts can detect at a glance the suspicious device in cluster 2, but clustering results do not give information about the changes and do not explain why this device is in a different cluster. Therefore symbolic description of every cluster will give analysts this valuable information to quickly narrow the scope, isolate the modified device and apply the correction measures.

Table 4 shows the description and explanation of the studied clusters after applying the DAU algorithm. There are two clusters and each cluster is represented by two rows in Table 4. For every cluster, the first row contains the value of common features and the second row contains how frequently that feature is present in the elements of that cluster. Analysts can select the degree of occurrence of the attributes to discard the less frequent ones. In Table 4 the most representative features are in bold type as they are common to more of the 80% of the elements of a cluster. Taking a closer look to the descriptions of both clusters, analysts can detect that the OS has not been altered, as all attributes of clusters 1 and 2 are related to the same OS. The real OS of all devices in that lab was XP with Service Pack 2, and this feature is in both clusters. On the other hand, analysts can easily notice that cluster 2 contains high number of filtered ports (value = 2). A filtered port means that a firewall, filter or some other network obstacle is blocking the port and preventing the system from determining whether it is open. Then, analysts can promptly solve this challenge without having to analyze vulnerabilities of all lab devices.

**Table 4**
Symbolic description of clusters containing lab devices after applying DAU.

| Cluster | Ports | | | | | | | | W2000 | | XP | | 2003 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7,9, 13,17, 19,21, 25 | 80 | 443 | 135 | 139 | 445 | 781–807 | 904–930 | WS_SP4 (%) | S_SP2 (%) | SP1 (%) | SP2 (%) | Standard (%) | Enterp. (%) |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | – | – | 67 | 67 | 67 | 70 | 70 | 70 |
| | 16% | 33% | 33% | 83% | 100% | 100% | | | 100 | 100 | 100 | 100 | 100 | 100 |
| 2 | – | – | – | 1 | 1 | 1 | 2 | 2 | 67 | 67 | 67 | 67 | 70 | – |
| | – | – | – | 100% | 100% | 100% | 100% | 100% | 100 | 100 | 100 | 100 | 100 | – |

The improvement of DAU allows detecting common ports, not only for all devices of a cluster, but also for a group of devices. For example, AU would have not given information to analysts about the port 135, as this attribute is common to 83% of the elements of cluster 1, but not common to the 100% of the elements of that cluster. When using DAU, information related about that port is also shown in the description, which allows corroborating that there have been no significant changes related to this attribute because it is present in both clusters. DAU description also shows that cluster 1 contains some devices that have altered services related to ports 7, 9, 13, 17, 19, 21, 25, 80 and 443. These changes have been not so relevant to separate those devices in another cluster, but give valuable information to analysts as all devices of cluster 1 should contain exactly the same features with the same values. Consequently analysts can also detect that some devices in cluster 1 have been modified as well.

## 7. Conclusions and further work

Network security audits provide large amount of data to process. Unsupervised learning clustering offers a valuable solution to automate the analysis of this data in order to identify behavior patterns and detect unauthorized modifications in a network. However, this data may contain uncertainty and partial knowledge. In this environment, soft computing can be useful to extract implicit, previously unknown and potentially useful information to security analysts. This paper has shown that clustering, and especially SOM, is a useful technique to aid analysts in this laborious task. Different clustering methods have been included in *Analia* to process data after a network security test. Experimentation has shown that SOM results provide clusters that contain devices with similar open ports, operating systems and vulnerabilities.

With the aim of giving a better explanation of clustering results to security analysts, *Analia* has been also improved with a new method to implement descriptions. The method presented in this paper is based on the anti-unification technique. We have proposed a new version of AU called DAU. This proposal considers not only the attributes common to all elements in a cluster, but also the attributes common to most of them. DAU provides *Analia* with more detailed explanations of every obtained cluster. This information is also useful to analysts to detect a misuse or misconfiguration in part of the devices from a cluster. Explanations are described using vocabulary comprehensible to security analysts, providing a useful method to justify clustering results and obtain helpful conclusions from data. As a result, the combination of clustering with DAU gives an effective system that takes advantage of the best features of both techniques.

Further work considers a deeper study of the explanations obtained from clusters in order to refine them and include the knowledge of security experts whenever possible. Clustering and explanations are now based on main features, e.g., open ports, operating systems and security information. On the other hand, *Consensus* stores more data regarding tested devices. Then, other knowledge representations that include more attributes will be analysed. This study will comprise the adaption of the existent clustering algorithms or other clustering methods to facilitate the management of these new attributes.

## Acknowledgments

This work has been partially supported by the MCYT-FEDER project called MID-CBR (TIN2006-15140-C03-01 and TIN2006-15140-C03-03) and by the *Generalitat de Catalunya* under Grants 2005SGR-302 and 2005-SGR-00093. We would also like to thank *La Salle-Universitat Ramon Llull* for the support to our research group.

## References

[1] A. Aamodt, E. Plaza, Case-based reasoning: foundations issues, methodological variations, and system approaches, IA Communications 7 (1994) 39–59.
[2] E. Armengol, E. Plaza, Bottom-up induction of feature terms, Machine Learning 41 (2000) 259–294.
[3] E. Armengol, E. Plaza, Symbolic explanation of similarities in CBR, Computing and Informatics 25 (2006) 1001–1019.
[4] M. Belanger, J. Martel, An automated explanation approach for a decision support system based on MCDA, in: AAAI Fall Symposium on Explanation-aware Computing, AAAI Press, 2005, pp. 21–34.
[5] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[6] E. Bloedorn, L. Talbot, D. DeBarr, Data Mining Applied to Intrusion Detection: MITRE Experiences, Machine Learning and Data Mining for Computer Security, Springer, Berlin, 2006.
[7] G. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer Vision, Graphics and Image Processing 37 (1987) 54–115.
[8] J. Cassens, Knowing what to explain and when, in: P. Gervás, K. Gupta (Eds.), Proceedings of ECCBR Workshops, 2004, pp. 97–104.
[9] P. Cheeseman, J. Stutz, Bayesian classification (autoclass): Theory and results, Advances in Knowledge Discovery and Data Mining, 1996, pp. 153–180.
[10] W. Cheetham, S. Shiu, R. Weber, Soft case-based reasoning, The Knowledge Engineering, 2005, pp. 1–4.
[11] G. Corral, A. Zaballos, X. Cadenas, A. Grané, A distributed security system for an intranet, in: 39th IEEE Carnahan Conference on Security Technology, 2005, pp. 291–294.
[12] G. Corral, E. Armengol, A. Fornells, E. Golobardes, Data security analysis using unsupervised learning and explanations, in: Innovations in Hybrid Intelligent Systems, Advanced in Soft Computing, Springer, Berlin, 2007, pp. 112–119.
[13] G. Corral, A. Fornells, E. Golobardes, J. Abella, Cohesion factors: improving the clustering capabilities of Consensus, in: Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Sciences, vol. 4224, Springer, Berlin, 2006, pp. 488–495.
[14] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Learning 4 (1979) 224–227.
[15] R. Davis, B.G. Buchanan, E.H. Shortliffe, Production systems as a representation for a knowledge-based consultation program, Artificial Intelligence 8 (1977) 15–45.
[16] G. Dejong, Explanation-based learning, in: A. Tucker (Ed.), Computer Science Handbook, CRC, Boca Raton, 2004.
[17] L. DeLooze, Classification of computer attacks using a self-organizing map, in: Proceedings of IEEE Workshop on Information Assurance, 2004, pp. 365–369.
[18] M. Depren, M. Topallar, Network-based anomaly intrusion detection system using SOMs, in: IEEE 12th Signal Processing and Communications Applications, 2004, pp. 76–79.

[19] D. Doyle, P. Cunningham, D. Bridge, Explanation oriented retrieval, in: Advances in Case-based Reasoning, 7th ECCBR, Lecture Notes in Computer Science, vol. 3155, Springer, 2004, pp. 157–168.
[20] J. Dunn, Well separated clusters and optimal fuzzy partitions, Journal of Cybernetics 4 (1974) 95–104.
[21] E. Eskin, A. Arnold, M. Preraua, L. Portnoy, S. Stolfo, A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, Applications of Data Mining in Computer Security, Kluwer Academic Publishers, Dordrecht, 2002.
[22] A. Fornells, E. Golobardes, D. Vernet, G. Corral, Unsupervised case memory organization: analysing computational time and soft computing capabilities, in: 8th European Conference on CBR, LNAI, vol. 4106, Springer, 2006, pp. 241–255.
[23] M. Gu, A. Aamodt, X. Tong, Component Retrieval Using Conversational Case-based Reasoning, Intelligent Information Processing II, Springer, Berlin, 2004.
[24] J. Hartigan, M. Wong, A *k*-means clustering algorithm, Applied Statistics 28 (1979) 100–108.
[25] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
[26] S. Kaski, J. Kangas, T. Kohonen, Bibliography of Self-organizing Map (SOM) Papers: 1981–1997, ⟨http://www.cis.hut.fi/research/refs/⟩, 1998.
[27] T. Kohonen, Self-organizing Maps, third ed., Springer, Berlin, 2000.
[28] K. Leung, C. Leckie, Unsupervised anomaly detection in network intrusion detection using clusters, in: Proceedings of 28th Australasian CS Conference, vol. 38, 2005.
[29] L. McGinty, B. Smyth, On the role of diversity in conversational recommender systems, in: 5th International Conference on Case-based Reasoning, 2003, pp. 276–290.
[30] D. McSherry, Explanation in recommender systems, Artificial Intelligence Review 24 (2005) 179–197.
[31] R. Michalski, Knowledge acquisition through conceptual clustering: a theoretical framework and algorithm for partitioning data into conjunctive concepts, International Journal of Policy Analysis and Information Systems 4 (1980) 219–243.
[32] C. Nugent, P. Cunningham, A case-based explanation system for black-box systems, Artificial Intelligence Review 24 (2) (2005) 163–178.
[33] M. Oja, S. Kaski, T. Kohonen, Bibliography of Self-organizing Map (SOM) Papers: 1998–2001, ⟨http://www.cis.hut.fi/research/refs/⟩, 2003.
[34] T. Peltier, Managing a Network Vulnerability Assessment, CRC Press, Boca Raton, 2003.
[35] D. Pelleg, A. Moore, *X*-means: extending *K*-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conference of Machine Learning, Morgan Kaufmann, 2000, pp. 727–734.
[36] M. Ramadas, S. Ostermann, B. Tjaden, Detecting anomalous network traffic with SOMs, in: 6th Symposium on Recent Advances in Intrusion Detection, vol. 2820, 2003, pp. 36–54.
[37] P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[38] R. Stepp, R. Michalski, Conceptual clustering: inventing goal-oriented classifications of structured objects, in: Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, 1986, pp. 471–498.
[39] S. Zanero, S. Savaresi, Unsupervised learning techniques for an intrusion detection system, in: Proceedings of ACM Symposium on Applied computing, 2004, pp. 412–419.

**Guiomar Corral** received her M.Sc. degree in electronic engineering from Universitat Ramon Llull in 1998. She is a member of the Research Group in Intelligent Systems (GRSI) of Enginyeria i Arquitectura La Salle—Universitat Ramon Llull and an associate professor since 1999. Her research interests are mainly focused on telematics, security on data networks, machine learning, clustering, soft computing and artificial intelligence applied in network security.

**Eva Armengol** received her Ph.D. in Computer Science by the Universitat Politecnica de Catalunya in 1997. She is now a permanent scientist of the Artificial Intelligence Institute (IIIA). Her research has been mainly focused on knowledge representation, machine learning and case-based reasoning. Currently she focuses her research on how to explain the results of learning methods.

**Albert Fornells** received his Ph.D. degree in Computer Science by Universitat Ramon Llull, Spain, in 2007. He is a member of the Research Group in Intelligent Systems from the university since 2000, and an associate professor since 2003. His research interests include data mining and knowledge discovery, self-organizing maps, case-based reasoning, soft computing, soft case-based reasoning and artificial intelligence applied in health care (medicine).

**Elisabet Golobardes** received her Ph.D. in Computer Science by Universitat Ramon Llull in 1998. She is a member of the Research Group in Intelligent Systems (GRSI) of Enginyeria i Arquitectura La Salle—Universitat Ramon Llull since 1994 and titular professor since 2000. Her research interests are mainly focused on case-based reasoning, soft computing, clustering, machine learning, data mining and knowledge discovery, artificial intelligence applied in health care (medicine) and applied in network security.