A Case-Based Approach to Anomaly Intrusion Detection

Alessandro Micarelli and Giuseppe Sansonetti

Department of Computer Science and Automation
Artificial Intelligence Laboratory
Roma Tre University
Via della Vasca Navale, 79, 00146 Rome, Italy
{micarel,gsansone}@dia.uniroma3.it

Abstract. The architecture herein advanced finds its rationale in the visual interpretation of data obtained from monitoring computers and computer networks with the objective of detecting security violations. This new outlook on the problem may offer new and unprecedented techniques for intrusion detection which take advantage of algorithmic tools drawn from the realm of image processing and computer vision. In the system we propose, the normal interaction between users and network configuration is represented in the form of snapshots that refer to a limited number of attack-free instances of different applications. Based on the representations generated in this way, a library is built which is managed according to a case-based approach. The comparison between the query snapshot and those recorded in the system database is performed by computing the Earth Mover's Distance between the corresponding feature distributions obtained through cluster analysis.

1 Introduction

Intrusion Detection Systems (IDSs) have the objective of detecting attacks launched against computers or computer networks. Their classification is usually based on the *audit source location* and on the *general detection strategy*. With respect to the first criterion, IDSs are divided into *host-based* techniques if the input information they analyze consists of audit trails and/or system logs and *network-based* techniques if it consists of network packets. According to the second criterion, IDSs are classified as *misuse-based* or *anomaly-based* techniques. The former use attack descriptions (*signatures*) in order to analyze the sequence of events obtained from monitoring a given network and single computers connected to it. If a known attack pattern is detected, an alarm is triggered. These systems are usually efficient and generate a limited number of false detections, called *false positives*. The main drawback of these systems lies in their inability to detect unknown attacks, *i.e.*, attacks for which there exists no prior information in the system database.

The anomaly-based IDSs follow an approach which is complementary to the previous one. They are based on models of the *normal* behavior (*profiles*) of users and applications in order to detect anomalous activities which might provide an indication of an internal intrusion, launched by users attempting to abuse of their privileges, or of an external intrusion. The main advantage of this approach is the fact that it is capable of identifying unknown attacks. This advantage is however obtained at the price of a large number of false positives. Axelsson refers to it as "the limiting factor for the performance of an anomaly-based IDS" [2]. In addition to this, recent work [33,37] has shown that these systems are vulnerable to *mimicry attacks*, *i.e.*, attacks which aim at imitating normal activity, thereby avoiding identification by the system.

Nonetheless, we believe that the benefits offered by anomaly-based systems are such that a thorough critical analysis of the limits of the approaches advanced so far is needed in order to come up with adequate solutions. In particular, excluding some notable exceptions, most anomaly-based systems share these common characteristics:

1. They are based on a single feature, *i.e.*, they usually consider a single characteristic, based on which they assess the normality of a generic user-application interaction;
2. They have only one input since they examine only one data typology, relative either to the network or to a generic host and they do not propose the analysis of combined data;
3. The classification procedure, *i.e.*, the procedure whereby a generic event is considered part of an ongoing attack or not, once the relative anomaly score is known is trivial.

Concerning the first characteristic, most techniques used to date do not make appropriate tools available to take into account more than one element during the evaluation phase. It is therefore worthwhile to explore new techniques, inspired by different principles.

The rest of the paper is organized as follows. Section 2 outlines related work. Section 3 presents our intrusion detection system, in particular the case representation and the dissimilarity metric. Section 4 describes the experiments that were performed to evaluate the accuracy of the case-based classifier. Section 5 contains the conclusions and Section 6 discusses future directions of our research.

2 Related Work

In the field of Intrusion Detection, in addition to the traditional techniques used to date, various alternative solutions have recently been advanced which use, among others, *haptic technologies* [13], capable of generating tactile sensations, and *sonification techniques* [34], which make use of non-speech audio to convey information. *Visualization techniques* have also been proposed which operate the conversion from textual datasets to digital images [7,23,20]. There exist many advantages associated with this conversion. In particular, it has been observed that, from a physiological viewpoint, the interpretation of graphical images is a parallel process and, as such, it is much more efficient than reading textual information, which is an intrinsically serial process [22].

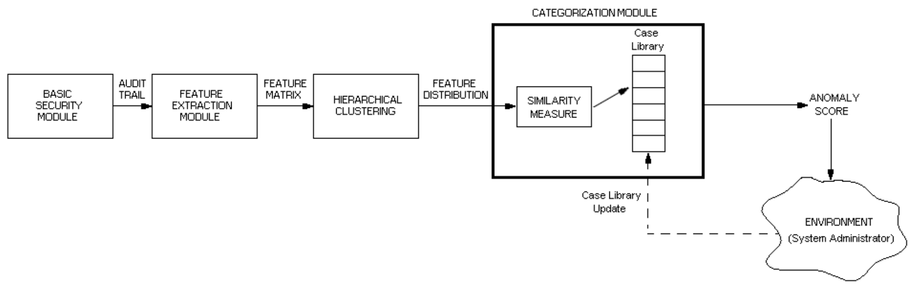


Fig. 1. The anomaly-detection system

Another major advantage is that a single image can convey several pieces of information simultaneously in a more structured and compact form than text [10]. There have been several contributions in the field of network-based visualization techniques, mainly aimed at representing relative performances and bandwidth usage in a graphical form [5,21].

Less attention has instead been devoted to Intrusion Detection. Among the early contributions, tools have been advanced to estimate the level of attack which a system being monitored undergoes [31,19]. Despite their usefulness, these tools only allow one to detect attacks which are already in progress, but they do not provide any proactive measure.

More recently, visual user interfaces have been devised which assist in the interpretation of data streams produced by IDSs [9,20,27,32]. If, on the one hand, these systems provide an important contribution, on the other hand, they can make the human interpretation of data easier but they do not replace it altogether.

To the best of our knowledge, there has been so far no contribution in automatic intrusion detection based on Image Processing and Computer Vision techniques. However, we believe that these fields have made an outstanding progress in providing useful tools in non-traditional application areas for these disciplines such as Intrusion Detection [12,30,17].

Much headway has been made since February 1992 when the National Science Foundation organized a workshop on *Visual Information Management Systems* in Redwood, California. The objective of this event was the identification of topic areas where to focus research aimed at designing and testing effective visual information management systems [24]. Such an interest was captured by the possibility to access large image databases where traditional query methods such as keywords and annotations cannot be used [6]. *Content-Based Image Retrieval* (CBIR) systems are nowadays largely in use. They exploit color, texture, shape information and spatial relations to represent and retrieve information [35]. Their large number and the excellent performances they can guarantee have inspired us to explore the use of such techniques in the arena of Intrusion Detection.

Concerning the use of Machine Learning techniques, several IDSs have resorted to them to improve their performance [18,8,3]. These systems can be grouped in two families: *rule-based* and *model-based* techniques. Even though these systems have been proven to be useful, they however suffer from the typical drawbacks of this kind of expert systems, *i.e.*, difficulties in the acquisition and representation of new knowledge.

Instead, *Case-Based Reasoning* (CBR) is a problem-solving paradigm which, rather than relying exclusively on general knowledge of the domain of interest or building associations through generalized relationships among problem descriptors and conclusions, it is capable of exploiting specific knowledge derived from situations (*cases*) already experienced and solved in the past [1].

In [14,11] a case-based reasoner (AUTOGUARD) for intrusion detection is presented. In AUTOGUARD, a translator model converts the low-level audit trail into high-level class representation of events. This information is recorded in the system as a collection of cases. In order to evaluate the similarity between the new case and every old case archived in the system library, the authors propose a fuzzy logic based approach. However, it is not clear if the design has been implemented altogether.

3 System Design

A block diagram representation of the system we have designed and implemented is shown in Figure 1. The input parameters are represented by the data obtained from monitoring computers connected to the network whereas the output parameter is the relative *anomaly score*. This value is given by the smallest value of dissimilarity obtained by comparing the input case with those stored in the database. This database is managed, queried, and updated according to modalities typical of the CBR approach.

It should be noted that the phase of relevance feedback is fundamental to keep the case record updated. In order for an input representation to be useful and, therefore, stored to optimize the system performance in case similar situations are encountered again, two requirements are necessary:

1. The environment (a term which also refers to human supervision, *e.g.*, the system administrator) has to confirm the system indications;
2. The input representation has to convey meaningful information, *i.e.*, in the database, there is no case capable of representing effectively the class the input snapshot belongs to.

Concerning the second objective, it is achieved using a second similarity threshold: in addition to an upper threshold (called *reliability threshold*) beyond which we can infer that the behavior being monitored is symptomatic of an attack underway, we have considered a lower threshold (called *identity threshold*) below which the input case is not kept. In other words, the input case is added to the knowledge base of the system, thereby assuming the characteristic of a profile, when its dissimilarity value with respect to all other cases archived in the library and relative to the same application is comprised between the two thresholds. This ensures that the cases, which are progressively added to the library, effectively reproduce a behavior not yet represented in the database. Therefore, they have to be recorded with the goal of optimizing the system performance in case a similar situation is encountered.

The need for carefully choosing the cases to keep stems from the need for optimizing the system resources, *i.e.*, memory support and processing time. Not only do these problems affect the system architecture herein proposed, but they also concern any case-based system. For this reason, they have been the object of research in the Artificial Intelligence (AI) community. There are several contributions suggesting memory models alternative to the simple flat memory. The interested reader is referred, for instance, to [25,38].

It should be noted that the domain expert possibility to intervene in the decision task is possible not only in the initial training phase of the system, but also during the verification phase for the classification response. The system is actually capable of acquiring knowledge also during its normal operation. The ease and quickness of the learning phase represent in fact some of the strong features of our case-based system.

In the following sections, we will analyze the key components of a typical case-based expert system, *i.e.*, the different case representations and the associated (dis)similarity metric.

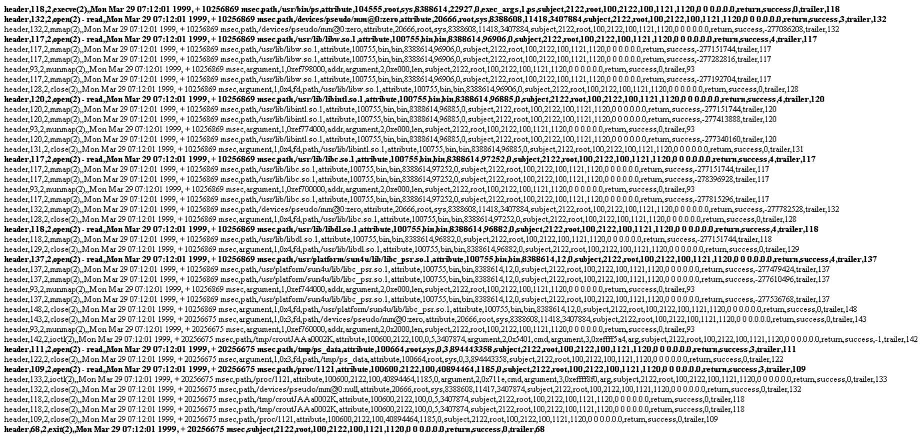
3.1 Case Representation

The fundamental assumption of the proposed architecture is the following: in order for a program to effectively damage the system being monitored, it has to interact with the operating system through system calls.

Various host-based approaches to anomaly detection have been proposed which build profiles from the sequences of system calls [16,36]. Specifically, these systems are based on models of the system call sequences generated by the applications during the normal operation of the system. In the detection phase, every sequence being monitored which is not compliant with the profiles previously recorded is deemed a part of an attack. Later work has, however, shown that it is possible for the intruder to avoid this kind of detection [33,37].

An effective solution thus advocates the exploitation of additional information drawn from the audit files. In [26], the authors observe that the output parameters and the arguments of the system calls can play an important role in the intrusion detection process. Based on these considerations, we have decided to consider this information in our representation. Concerning the output parameters, namely the return value and the error status, their use is straightforward since they are already available in a numeric format.

The issue is more complicated with the system call arguments. These arguments can be divided into four categories: *file name*, *execution parameter*, *user ID*, and *flag* [26]. The first two are of `string` type, the other of `integer` type.



```
header,118,2,execve(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/bin/ps,attribute,104555,root,sys,8388614,22927,0,exec_args,1,ps,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,118
header,132,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/devices/pseudo/mm@0:zero,attribute,20666,root,sys,8388608,11418,3407884,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,3,trailer,132
header,132,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/devices/pseudo/mm@0:zero,attribute,20666,root,sys,8388608,11418,3407884,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277086208,trailer,132
header,117,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libw.so.1,attribute,100755,bin,bin,8388614,96906,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,4,trailer,117
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libw.so.1,attribute,100755,bin,bin,8388614,96906,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277151744,trailer,117
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libw.so.1,attribute,100755,bin,bin,8388614,96906,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277282816,trailer,117
header,93,2,munmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0xef798000,addr,argument,2,0xe000,len,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,93
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libw.so.1,attribute,100755,bin,bin,8388614,96906,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277192704,trailer,117
header,128,2,close(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0x4,fd,path,/usr/lib/libw.so.1,attribute,100755,bin,bin,8388614,96906,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,128
header,120,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libintl.so.1,attribute,100755,bin,bin,8388614,96885,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,4,trailer,120
header,120,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libintl.so.1,attribute,100755,bin,bin,8388614,96885,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277151744,trailer,120
header,120,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libintl.so.1,attribute,100755,bin,bin,8388614,96885,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277413888,trailer,120
header,93,2,munmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0xef774000,addr,argument,2,0xe000,len,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,93
header,120,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libintl.so.1,attribute,100755,bin,bin,8388614,96885,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277340160,trailer,120
header,131,2,close(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0x4,fd,path,/usr/lib/libintl.so.1,attribute,100755,bin,bin,8388614,96885,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,131
header,117,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libc.so.1,attribute,100755,bin,bin,8388614,97252,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,4,trailer,117
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libc.so.1,attribute,100755,bin,bin,8388614,97252,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277151744,trailer,117
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libc.so.1,attribute,100755,bin,bin,8388614,97252,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-278396928,trailer,117
header,93,2,munmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0xef700000,addr,argument,2,0xe000,len,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,93
header,117,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libc.so.1,attribute,100755,bin,bin,8388614,97252,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277815296,trailer,117
header,132,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/devices/pseudo/mm@0:zero,attribute,20666,root,sys,8388608,11418,3407884,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277782528,trailer,132
header,128,2,close(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0x4,fd,path,/usr/lib/libc.so.1,attribute,100755,bin,bin,8388614,97252,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,128
header,118,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libdl.so.1,attribute,100755,bin,bin,8388614,96882,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,4,trailer,118
header,118,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/lib/libdl.so.1,attribute,100755,bin,bin,8388614,96882,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277151744,trailer,118
header,129,2,close(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0x4,fd,path,/usr/lib/libdl.so.1,attribute,100755,bin,bin,8388614,96882,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,129
header,137,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/platform/sun4u/lib/libc_psr.so.1,attribute,100755,bin,bin,8388614,12,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,4,trailer,137
header,137,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/platform/sun4u/lib/libc_psr.so.1,attribute,100755,bin,bin,8388614,12,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277479424,trailer,137
header,137,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/platform/sun4u/lib/libc_psr.so.1,attribute,100755,bin,bin,8388614,12,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277610496,trailer,137
header,93,2,munmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0xef744000,addr,argument,2,0xe000,len,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,93
header,137,2,mmap(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,path,/usr/platform/sun4u/lib/libc_psr.so.1,attribute,100755,bin,bin,8388614,12,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-277536768,trailer,137
header,148,2,close(2),,Mon Mar 29 07:12:01 1999, + 10256869 msec,argument,1,0x4,fd,path,/usr/platform/sun4u/lib/libc_psr.so.1,attribute,100755,bin,bin,8388614,12,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,148
header,143,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,argument,1,0x3,fd,path,/devices/pseudo/mm@0:zero,attribute,20666,root,sys,8388608,11418,3407884,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,143
header,93,2,munmap(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,argument,1,0xef760000,addr,argument,2,0x2000,len,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,93
header,142,2,ioctl(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/tmp/croutJAAa0002K,attribute,100600,2122,100,0,5,3407874,argument,2,0x5401,cmd,argument,3,0xeffff5a4,arg,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,-1,trailer,142
header,111,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/tmp/ps_data,attribute,100664,root,sys,0,3,894443358,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,3,trailer,111
header,122,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,argument,1,0x3,fd,path,/tmp/ps_data,attribute,100664,root,sys,0,3,894443358,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,122
header,109,2,open(2) - read,,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/proc/1121,attribute,100600,2122,100,40894464,1185,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,3,trailer,109
header,133,2,ioctl(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/proc/1121,attribute,100600,2122,100,40894464,1185,0,argument,2,0x711e,cmd,argument,3,0xeffff8f0,arg,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,133
header,132,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/devices/pseudo/mm@0:null,attribute,20666,root,sys,8388608,11417,3407874,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,132
header,118,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/tmp/croutJAAa0002K,attribute,100600,2122,100,0,5,3407874,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,118
header,118,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/tmp/croutJAAa0002K,attribute,100600,2122,100,0,5,3407874,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,118
header,109,2,close(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,path,/proc/1121,attribute,100600,2122,100,40894464,1185,0,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,109
header,68,2,exit(2),,Mon Mar 29 07:12:01 1999, + 20256675 msec,subject,2122,root,100,2122,100,1121,1120,0 0 0.0.0.0,return,success,0,trailer,68
```

Fig. 2. Application audit trail

In this preliminary version of our system, we have considered only the `string` type, for which it is possible to consider three models, namely, the length, the character distribution and the grammar inference. The length and character distribution models can be applied straightforwardly, since, with reference to the second, we are only interested in the profile generated by the frequency of occurrence of the characters independent of their type.

Concerning the grammar inference, *i.e.*, the inference of the argument grammar, two processing steps are necessary. In the first, each character is replaced by the token corresponding to its class; in the second, the possible repetitions of elements belonging to the same class are merged [26].

Regarding the classes, we have considered three main groups of characters, namely `lowercase` letters, `uppercase` letters, and `digits`. Characters which do not belong to any of these classes are considered to belong to new classes. A different numeric identifier is associated with each class. For instance, assuming the following class-identifier association:

N_1:	`lowercase` letter
N_2:	`uppercase` letter
N_3:	`digit`
N_4:	`slash`
$\dots$:	$\dots$

the string `/etc/usr/bin` is represented in terms of the these ten features:

$$N_4, N_1, N_4, N_1, N_4, N_1, 0, 0, 0, 0.$$

The input to the detection process is an ordered stream $X = \{x_1, x_2, \cdots\}$ of system call invocations representing the generic instance of an application. In

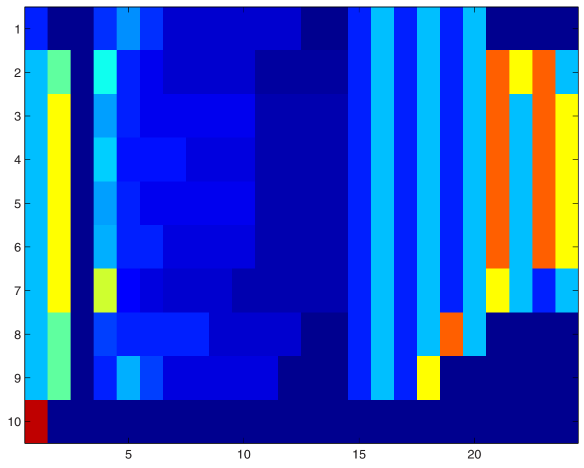


Fig. 3. Application snapshot

our system, based on the previous considerations, every system call invocation $x \in X$ is represented by means of the following features

$$< f_1^x, f_2^x, f_3^x, f_4^x, f_5^x, \cdots, f_{14}^x, f_{15}^x, \cdots, f_{24}^x >$$

where

f_1^x:	system call class
f_2^x:	return value
f_3^x:	error status
f_4^x:	argument length
$f_5^x, \cdots, f_{14}^x$:	argument character distribution
$f_{15}^x, \cdots, f_{24}^x$:	argument grammar inference

In particular, we have monitored the following six system calls: `execve()`, `chmod()`, `chown()`, `exit()`, `open()`, `setuid()`, since these are the only ones deemed potentially dangerous. In [4], Axelsson points out that "this logging method consumes as little system resources as comparable methods, while still being more effective."

In order to spot the sequence of system calls within an audit trail of a generic application, it is sufficient to find the audit record representing the `execve()` system call in which the path name of the application of interest appears and to record the *process ID* assigned to the process by the operating system. The system calls to represent are all those which appear one after the other up to the record relative to the `exit()` command terminating the process having the ID under consideration. From the whole sequence of system calls we have represented only the six described above. For these audit events we have converted in numeric features only the pieces of information relative to the output parameters (second and third columns) and to the arguments (remaining columns).

Based on these considerations, for instance, the instance of the `ps` application comprised of the 43 system calls shown in Fig. 2, is associated with an $m \times n$ matrix of features where m is the number of system calls of the following types `execve(), chmod(), chown(), exit(), open(), setuid()` among the overall 43, 10 in this case, whereas n is fixed and equal to 24, *i.e.*, to the number of attributes which we have decided to consider and whose corresponding values constitute the matrix entries.

Fig. 3 shows the representation obtained with Matlab by interpreting each matrix entry as an index in the RGB color space. We have thus obtained a snapshot representing the temporal behavior of the `ps` application to monitor; this can then be compared against the profiles relative to the `ps` application stored in the system.

Cluster Analysis. In order to compare system call sequences which may be rather different in terms of their structure and of their number, a cluster analysis is needed. In particular, we have used *Hierarchical Clustering* with the *Jaccard Distance* in order to calculate the distance between every pair of objects. This distance is defined as one minus the *Jaccard coefficient*, that is the percentage of nonzero coordinates that differ from each other. Given an $m \times n$ feature matrix X representing the generic instance of an application and made up of m $1 \times n$ row vectors x_1, x_2, ..., x_m, representing the relative system calls, the Jaccard distance between the row vectors x_r e x_s has the following expression:

$$d_{rs} = \frac{\#\left[(x_{rj} \neq x_{sj}) \wedge ((x_{rj} \neq 0) \vee (x_{sj} \neq 0))\right]}{\#\left[(x_{rj} \neq 0) \vee (x_{sj} \neq 0)\right]} \qquad (1)$$

where # is the cardinality.

Then we have set an *inconsistency coefficient* threshold to divide the objects in the hierarchical tree into clusters. This coefficient compares the height of a link in a cluster hierarchy with the average height of neighboring links. It is thus possible to identify the natural divisions in the dataset, but this involves a variable number of clusters for every instance of an application. Every instance of an application is represented by a set of a different number of clusters where each cluster is represented by the coordinates of its centroid and by a weight that, in the preliminary version of our system, is equal to the fraction of the distribution that belongs to that cluster (the procedure for assigning appropriate weights to the clusters will be one of the objectives of our future work). The information obtained in this way represents a case which is structured as a record comprising the following three fields:

- The first field is of `string` type and contains the name of the application to which it refers; it is obtained from the path of the relative `execve` system call;

- The second field is represented by an array of N records (where N is the number of clusters, a function of the threshold value for the inconsistency coefficient) having 24 fields of type `double`, which contain the values of the attributes represented and constitute the coordinates of the relative centroid;
- The third field is represented by an array of N values of type `double`, each expressing the weight of the corresponding cluster.

3.2 Dissimilarity Metric

Once the representation of an application instance has been generated according to the modalities discussed above, an appropriate dissimilarity metric has to be determined to compare the input case with those contained in the database.

Recently, the *Earth Mover's Distance* (EMD) [28] has been proposed to evaluate distribution dissimilarities. The EMD is based on the minimum cost associated with the transformation of one distribution into the other. In the case of Content-Based Image Retrieval, it has been proven to be more robust than the histogram-based techniques, since it is able to handle also representations with variable length. When used to compare distributions with the same overall mass, it can be readily shown that it is a real metric [29], which allows the use of more efficient data structures and query algorithms.

The EMD enables us to evaluate the dissimilarity between two multi-dimensional distributions. In our architecture, the two distributions are represented by two sets of weighted clusters that capture them. The clusters of any distribution can be in any number and the sum of their weights can be different than the sum of weights of the other distribution. This is the reason why a smaller sum appears at the denominator of the expression of the EMD. In order to calculate the EMD in some feature space, a distance measure (called *ground distance*) between single features must be defined.

The computation of the EMD value can be performed by solving the following linear programming problem: let X denote the distribution of the input instance of an application with m clusters,

$$X = \{(x_1, w_{x_1}), (x_2, w_{x_2}), \cdots, (x_m, w_{x_m})\} \tag{2}$$

where x_i represents the generic cluster and w_{x_i} the relative weight, and let Y denote the distribution of the generic instance of the same application in the archive of cases with n clusters

$$Y = \{(y_1, w_{y_1}), (y_2, w_{y_2}), \cdots, (y_n, w_{y_n})\} \tag{3}$$

Let $D = [d_{ij}]$ denote the ground distance matrix, d_{ij} being the ground distance between clusters x_i and y_j. The objective is to calculate the value of the flow $F = [f_{ij}]$ that minimizes the overall cost

$$WORK(X, Y, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij} \tag{4}$$

subject to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \tag{5}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{x_i} \qquad 1 \leq i \leq m \tag{6}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{y_j} \qquad 1 \leq j \leq n \tag{7}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = min\left(\sum_{i=1}^{m} w_{x_i}, \sum_{j=1}^{n} w_{y_j}\right) \tag{8}$$

Once we have calculated the value of the flow that solves the above equations, the EMD has the following expression

$$EMD(X,Y) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{9}$$

4 Empirical Evaluation

In order to evaluate the accuracy of our case-based IDS, we performed experimental runs divided in a first training phase and in a second testing phase. During the training phase, a database of instances of every application was built, which represented normal behavior. The testing phase then ensued.

For the experiments we used the 1999 MIT Lincoln Lab Intrusion Detection Evaluation Data [15]. In particular, we employed data of two attack-free weeks (First Week and Third Week) to train the system and data of two other weeks (Fourth Week and Fifth Week) to test the ability of the proposed architecture to correctly classify applications with attacks and applications associated with the users' normal behavior. For some of the attacks in the evaluation data there is no evidence in the Solaris Basic Security Module (BSM) log, so we were not interested in them. Among the visible attacks in the BSM audit trail some are *policy violations*, in which the intruder tried to exploit possible system configuration made by the administrator. We did not try to detect this class of attacks with our system but we plan on performing this test in the future work. In particular, we were interested in detecting attacks based on *buffer overflow vulnerabilities*.

In our simulations, a value of 0.9 was chosen for the threshold of the inconsistency coefficient. As a distance for clustering, we have used the Jaccard distance whereas for the computation of the EMD we have chosen the Euclidean distance as the ground distance.

In particular, we have carried out two different experimental runs. In the first experiment we stored in the library case all the 117 instances of `eject`, `fdformat`, `ffbconfig` and `ps` applications encountered in the training phase.

Table 1. Experimental Results

	Total	With attack	Identified	False Alarms
eject	9	3	3	0
fdformat	9	6	6	0
ffbconfig	2	2	2	0
ps	315	14	14	0
	335	25	25	0

We have considered these four applications since these are the only ones subject to attack in the Lincoln Laboratory database. We have then tested the system by using a value of 5 for the reliabilty threshold: the input application has been compared with all the instances archived in the library and relative to the same application. If the minimum value obtained was lower than the threshold, the application was labeled attack-free, otherwise it was classified as containing an attack.

In the second experiment we started with an initially empty database. Every training input application was analyzed through hierarchical clustering and compared to all existing entries in the case memory. If a distribution was found in the database that was similar enough, *i.e.*, below the identity threshold set to 0.5, according the EMD similarity metric, this new case was discarded, because it was already adequately represented in the database. Otherwise, the distribution (clusters with their weights) that corresponded to the new input was included into the database. After the training phase, the library contained only 19 cases. A testing phase was then carried out by choosing the same parameters as those of the previous session.

The results obtained after the two experimental sessions are collected in Table 1. The fact that we have obtained the same values after the two testing runs confirms that recording only one case for each typology of situation encountered in the training phase, with the objective of improving the computational efficiency of the system, does not have any effect altogether on the system performance in terms of classification.

Concerning the experimental results, we did not obtain any false positives by testing the system with 335 instances of input applications and all 25 applications containing attacks have been correctly identified.

5 Conclusions

In this contribution, we have presented a case-based anomaly detection system which was inspired by the interpretation in the form of snapshots of system call sequences obtained from the log of the C2 BSM of a Solaris workstation and relative to different instances of applications. This allowed us to resort to Image Processing and Computer Vision techniques, in particular to methodologies drawn from Content-Based Image Retrieval for the implementation of our system. These techniques, together with a CBR approach in the management of the knowledge base and with a representation of the cases based on the information relative to output parameters and arguments of the system calls, enabled us to obtain no false positives, even with a limited number of cases in the library. In particular, it was possible to distinguish the 25 instances of applications affected by attacks from the 310 relative to the normal behavior of the system with very high accuracy. This was confirmed by the appreciable differences among the EMD values relative to the corresponding feature distributions.

Obtaining a null number of false positives represents a very important result, in consideration of the fact that achieving a small number of false positives constitutes one of the most difficult objectives of any anomaly detection system.

Furthermore, the possibility to intervene on different parameters of the classification procedure (inconsistency coefficient threshold, reliability threshold, identity threshold, *etc.*) allows one to conveniently change the sensitivity of the system, thereby increasing the probability to identify also the so-called *mimicry attacks*.

The procedure we have advanced has therefore allowed us to fully exploit the salient features of the user-network configuration interaction, enabling the accurate distinction between attacks and events associated with the normal behavior of the system.

6 Future Work

There are several research thrusts that we intend to pursue in the near future. First of all, we will focus our efforts on the clustering procedure, particularly on the weight assignment procedure. Even though the experimental results we have obtained are satisfactory, we intend to take into account other factors, such as the semantic difference between the various features and the presence of outliers obtained from monitoring the host.

We will continue our experimental evaluation of the system performance, using new benchmarks, in order to check its capability of recognizing also new classes of attacks in addition to buffer overflows already identified. In particular, we will tackle the so-called *policy violations*, which to not allow the intruders to directly upgrade their privileges, but have the objective of gaining classified information in order to exploit possible erroneous configurations of the system administrator. This class of attacks thus contain intrusions which do not exploit actual system flaws and turn out to be not easily detectable, since the intruders have access to classified information through the normal, although unintentional, behavior of the system. In order to achieve this goal, besides working on clustering, it is necessary to further develop the modalities for representing the cases, taking into account new models based on the information contained in the audit trails, such as, for instance, *execution parameter*, *user ID* and *flag*.

Another objective of our future research will be the integration of profiles with signatures relative to known attacks. Last but not least, we will work on the realization of a network-based version of our intrusion detection system, in order to realize a combined analysis of the data obtained from monitoring the whole network configuration.

Acknowledgements

We would like to thank Dick Kemmerer, Giovanni Vigna, and Luca Lucchese for the opportunity to work at the Reliable Software Laboratory, Computer Science Department, University of California, Santa Barbara (CA), USA, and at the Projects Laboratory, School of Electrical and Computer Science, Corvallis (OR), USA.

References

1. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AICOM **7**(1), 39–59 (1994)
2. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy. In: Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1999, pp. 1–7. ACM Press, New York (1999)
3. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University (March 2000)
4. Axelsson, S., Lindqvist, U., Gustafson, U., Jonsson, E.: An Approach to UNIX Security Logging. In: Proceedings of the 21st NIST-NCSC National Information Systems Security Conference, Crystal City, VA, October 1998, pp. 62–75 (1998)
5. Becker, R., Eick, S.G., Wilks, A.: Visualizing Network Data. IEEE Transactions on Visualization and Computer Graphics **1**(1), 16–28 (1995)
6. Del Bimbo, A.: Visual Information Retrieval. Morgan Kaufmann Publishers, Inc. San Francisco, CA (1999)
7. Couch, A.: Visualizing Huge Tracefiles with Xscal. In: 10th Systems Administration Conference (LISA '96), Chicago, IL, October 1996, pp. 51–58 (1996)
8. Debar, H., Dacier, M., Wespi, A.: Towards a Taxonomy of Intrusion Detection Systems. Computer Networks 31(8), 805–822 (1999)
9. Erbacher, R.: Visual Traffic Monitoring and Evaluation. In: Proceedings of the Second Conference on Internet Performance and Control of Network Systems, Denver, CO, August 2001, pp. 153–160 (2001)
10. Erbacher, R., Frincke, D.: Visualization in Detection of Intrusions and Misuse in Large Scale Networks. In: Proceedings of the International Conference on Information Visualization '00, London, UK, July 2000, pp. 294–299 (2000)
11. Esmaili, M., Safavi-Naini, R., Balachandran, B.M.: AUTOGUARD: A Continuous Case-Based Intrusion Detection System. In: Proceedings of the 20th Australasian Computer Science Conference (1997)
12. Smeulders, A.W., et al.: Content-Based Image Retrieval at the End of the Early Years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(12), 1349–1380 (2000)
13. Nyarko, K., et al.: Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integration. In: Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Orlando, FL (2002)
14. Esmaili, M., et al.: Case-Based Reasoning for Intrusion Detection. In: Proceedings of the 12th Annual Computer Security Applications Conference, San Diego, CA (1996)

15. Lippmann, R.P., et al.: Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. In: Proceedings of Recent Advances in Intrusion Detection, Toulouse, France, pp. 162–182 (2000)
16. Forrest, S.: A Sense of Self for UNIX Processes. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pp. 120–198. IEEE Computer Society Press, Los Alamitos (1996)
17. Forsyth, D., Ponce, J.: Computer Vision: A Modern Approach. Prentice-Hall, Inc., Upper Saddle River, NJ (2003)
18. Frank, J.: Artificial Intelligence and Intrusion Detection: Current and Future Directions. In: Proceedings of the 17th National Computer Security Conference, Washington, D.C., pp. 22–33 (1994)
19. Frincke, D., Tobin, D., McConnell, J., Marconi, J., Polla, D.: A Framework for Cooperative Intrusion Detection. In: Proceedings of the 21th National Information Systems Security Conference, Crystal City, VA, October 1998, pp. 361–373 (1998)
20. Girardin, L., Brodbeck, D.: A Visual Approach for Monitoring Logs. In: Proceedings of the Second Systems Administration Conference (LISA XII), Boston, MA, October 1998, pp. 299–308 (1998)
21. He, T., Eick, S.G.: Constructing Interactive Visual Network Interfaces. Bells Labs Technical Journal 3(2), 47–57 (1998)
22. Hendee, W., Wells, P.: The Perception of Visual Information. Springer, Heidelberg (1994)
23. Hughes, D.: Using Visualization in System and Network Administration. In: Proceedings of the 10th Systems Administration Conference (LISA '96), Chicago, IL, October 1996, pp. 59–66 (1996)
24. Jain, R.: Proceedings of US NSF Workshop Visual Information Management Systems (1992)
25. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann Publishers, Inc., San Mateo, CA (1993)
26. Kruegel, C., Mutz, D., Valeur, F., Vigna, G.: On the Detection of Anomalous System Call Arguments. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 326–343. Springer, Heidelberg (2003)
27. Mizoguchi, F.: Anomaly Detection Using Visualization and Machine Learning. In: Proceedings of the 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'00), Gaithersburg, MD, March 2000, pp. 165–170 (2000)
28. Rubner, Y., Tomasi, C., Guibas, L.J.: A Metric for Distributions with Applications to Image Databases. In: Proceedings of the IEEE International Conference on Computer Vision, Bombay, India, January 1998, pp. 59–66. IEEE Computer Society Press, Los Alamitos (1998)
29. Rubner, Y., Tomasi, C., Guibas, L.J.: The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision 28(40), 99–121 (2000)
30. Shapiro, L.G., Stockman, G.C.: Computer Vision. Prentice-Hall, Inc., Upper Saddle River, NJ (2001)
31. Snapp, S.: DIDS (Distributed Intrusion Detection System): Motivation, Architecture and An Early Prototype. In: Proceedings of the National Information Systems Security Conference, Washington, D.C., October 1991, pp. 167–176 (1991)
32. Takada, T., Koike, H.: Tudumi: Information Visualization System for Monitoring and Auditing Computer Logs. In: Proceedings of the 6th International Conference on Information Visualization (IV'02), London, England, July 2002, pp. 570–576 (2002)

33. Tan, K., Killourhy, K., Maxion, R.: Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, Springer, Heidelberg (2002)
34. Varner, P.E., Knight, J.C.: Security Monitoring, Visualization, and System Survivability. In: 4th Information Survivability Workshop (ISW-2001/2002) Vancouver, Canada (March 2002) (2002)
35. Veltkamp, R.C., Tanase, M.: Content-Based Image Retrieval Systems: A Survey. Technical Report 2000-34, UU-CS, Utrecht, Holland (October 2000)
36. Wagner, D., Dean, D.: Intrusion Detection via Static Analysis. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pp. 40–47. IEEE Computer Society Press, Los Alamitos (2001)
37. Wagner, D., Soto, P.: Mimicry Attacks on Host-Based Intrusion Detection Systems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, D.C., pp. 255–264. ACM Press, New York (2002)
38. Watson, I.: Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann Publishers, Inc., San Francisco (1997)