Mining Common Outliers for Intrusion Detection

Goverdhan Singh, Florent Masseglia, Céline Fiot, Alice Marascu, Pascal Poncelet

Mining Common Outliers for Intrusion Detection

Goverdhan Singh and Florent Masseglia and Céline Fiot and Alice Marascu and Pascal Poncelet

Abstract Data mining for intrusion detection can be divided into several sub-topics, among which unsupervised clustering (which has controversial properties). Unsupervised clustering for intrusion detection aims to i) group behaviours together depending on their similarity and ii) detect groups containing only one (or very few) behaviour(s). Such isolated behaviours seem to deviate from the model of normality; therefore, they are considered as malicious. Obviously, not all atypical behaviours are attacks or intrusion attempts. This represents one drawback of intrusion detection methods based on clustering. We take into account the addition of a new feature to isolated behaviours before they are considered malicious. This feature is based on the possible repeated occurrences of the bahaviour on many information systems. Based on this feature, we propose a new outlier mining method which we validate through a set of experiments.

Goverdhan Singh
INRIA, 2004 route des lucioles - BP 93, FR-06902 Sophia Antipolis e-mail: g.singh@iitg.ernet.in

Florent Masseglia
INRIA, 2004 route des lucioles - BP 93, FR-06902 Sophia Antipolis e-mail: Florent.Masseglia@sophia.inria.fr

Céline Fiot
INRIA, 2004 route des lucioles - BP 93, FR-06902 Sophia Antipolis e-mail: celine.fiot@gmail.com

Alice Marascu
INRIA, 2004 route des lucioles - BP 93, FR-06902 Sophia Antipolis e-mail: Alice.Marascu@sophia.inria.fr

Pascal Poncelet
LIRMM UMR CNRS 5506, 161 Rue Ada, 34392 Montpellier Cedex 5, France e-mail: poncelet@lirmm.fr

1 Introduction

Intrusion detection is a very important topic of network security and has received much attention [Lee and Stolfo, 1998, Dokas et al., 2002, Lazarevic et al., 2003, Patcha and Park, 2007] since potential cyber threats make the organizations vulnerable. *Intrusion Detection Systems (IDS)* are intended to protect information systems against intrusions and attacks and are traditionally based on the signatures of known attacks [Roesch, 1998, Barbara et al., 2001]. Therefore, new kinds of attacks have to be added to the signature list regularly. The main drawback is that in case of an emerging attack (the recent discovery of a new security hole for instance), the IDS will ignore it since this new attack has not been listed yet in the signature database.

Protecting a system against new attacks, while keeping an automatic and adaptive framework is an important topic in this domain. One solution to this problem can be based on data mining tools. Data mining tools have been used to provide IDS with more adaptive detection approaches of cyber threats [Dokas et al., 2002, Bloedorn et al., 2001, Wu and Zhang, 2003]. Among these data mining approaches, predictive models are built to improve the database of signatures used by existing IDS [Wu and Zhang, 2003]. Other ones, whose category this chapter refers to, make use of data mining to detect anomalies from which the intrusions are deduced [Lazarevic et al., 2003, Eskin et al., 2002, Chimphlee et al., 2005]. The overall principle is generally to build clusters (or classes) of usage and, afterwards, to find the outliers (*i.e.* events that do not belong to any class or cluster corresponding to a normal usage). Actually, outlier detection aims to find records that deviate significantly from a well-defined notion of normality. It has a wide range of applications, such as fraud detection for credit card [Aleskerov et al., 1997], health care [Spence et al., 2001], cyber security [Ertoz et al., 2004] or safety of critical systems [Fujimaki et al., 2005]. However, the main drawback of detecting intrusions by means of anomaly (outliers) detection is the high rate of false alarms. In both cases (building a model or detecting outliers) an alarm can indeed be triggered because of a new kind of usages that has never been seen before; so it is considered abnormal. Considering the large amount of new usage patterns emerging in the Information Systems, even a weak percentage of false positive gives a very large amount of spurious alarms that would be overwhelming for the analyst. Reducing the rate of false alarms is thus crucial for a data mining based intrusion detection system in a real-world environment.

Therefore, the goal of this chapter is to propose an intrusion detection algorithm based on the analysis of usage data coming from multiple partners in order to reduce the number of false alarms. Our main idea is that a new usage is likely to be related to the context of the information system on which it occurs (so it should only occur on this system). On the other hand, when a new security hole has been found on a system, the hackers would use it in as many information systems as possible. Thus a new anomaly occurring on two (or more) information systems is rather an intrusion attempt than a new kind of usage. Let us consider A_x, an anomaly detected in the usage of web site S_1 corresponding to a php request on the staff directory for a new employee: John Doe, who works in room 204, floor 2, in the R&D department.

The request has the following form: `staff.php?FName=John\&LName=Doe`
`\&room=204\&floor=2\&Dpt=RD`. This new request, due to the recent re-
cruitment of John Due in this department, should not be considered as an attack.

Let us now consider A_y, an anomaly corresponding to a real intrusion. A_y is
caused by a security hole of the system (for instance a php vulnerability) and might,
for instance, look like: `staff.php?path=../etc/passwd%00`. In this re-
quest, one can see that the parameters are not related to the data accessed by the
php script, but rather to a security hole discovered on the *staff* script. If two or more
firms use the same script (say, a directory requesting script bought to the same soft-
ware company) then the usage of this security hole is certainly repeated from one
system to another and the request having parameter `path=../etc/passwd%00`
will be the same for all the victims.

We propose to provide the end-user with a method that has only one parameter: n,
the number of desired alarms. Based on the analysis of the usage data coming from
the different partners, our algorithm will detect n common outliers they share. Such
common outliers are likely to be true attacks and will trigger an alarm. In a real-
world application of this technique, privacy preserving will be a major issue in order
to protect partners' data. We focus on clustering and outlier detection techniques in
a distributed environment. However, privacy issues in our framework are currently
being studied.

The chapter is organized as follows. In Section 2 we present the motivation of this
approach and our general framework. Section 3 gives an overview of existing works
in this domain. Section 4 presents COD, our method for detecting outliers and trig-
gering true alarms. Eventually, our methods is tested through a set of experiments
in Section 5 and Section 6 gives the conclusion.

2 Motivation and General Principle

In this section, we present the motivation of our work, based on the main drawbacks
of existing anomaly-based methods for intrusion detection and we propose COD, a
new algorithm for comparing the anomalies on different systems.

2.1 Motivation

Anomaly-based IDS [Eskin et al., 2002, Chimphlee et al., 2005] can be divided into
two categories: semi-supervised and unsupervised. Semi-supervised methods use a
model of "normal" behaviours on the system. Every behaviour that is not considered
as normal is an anomaly and should trigger an alarm. Unsupervised methods do
not use any labelled data. They usually try to detect outliers based on a clustering
algorithm.

Obviously, *anomaly-based IDS will suffer from a very high number of false alarms since a new kind of behaviour will be considered as an anomaly (and an attack)*. Actually, anomalies are usually extracted by means of outlier detection, which are records (or sets of records) that significantly deviate from the rest of the data. Let us consider, for instance, a dataset of 1M navigations collected during one week on the Web site of a company (say, a search engine). In this case, a false alarm rate of 2% represents $20,000$ alarms that could be avoided. Reducing the number of false alarms is linked to the detection rate. However, the primary motivation of our work is to lower the rate of false alarms. We propose to improve the results of unsupervised IDS by means of a collaborative framework involving different network-based systems. Section 3 gives an overview of existing IDS based on the principles presented above and on the existing collaborative IDS. However, to the best of our knowledge, our proposal is the first unsupervised IDS using the common anomalies of multiple partners in order to detect the true intrusion attempts. The main idea of our proposal is that multiple partners do not share the same data, but they share the same systems (the Web server can be Apache or IIS, the data server can run Oracle, the scripts accessing the data can be written with PHP or CGI, etc). When a security hole has been found in one system (for instance, a php script with specific parameters leading to privileged access to the hard drive), then this weakness will be the same for all the partners using the same technology. Our goal is to reduce the rate of false alarm based on this observation, as explained in section 2.2

2.2 General Principle

In this chapter we present COD (Common Outlier Detection) a framework and algorithm intended to detect the outliers shared by at least two partners in a collaborative IDS. Outliers are usually small clusters. Some outlier detection methods are presented in section 3. As explained in section 2.1 the main drawback of clustering-based IDS is that they obtain a list of outliers containing both normal atypical usages and real intrusions; so the real intrusions are not separated from the normal atypical behaviors. Our goal is to compare such outlier lists from different systems (based on a similar clustering, involving the same distance measure). If an outlier occurs for at least two systems, then it is considered as an attack. COD is based on following the assumptions:

- An intrusion attempt trying to find a weakness of a script will look similar for all the victims of this attack.
- This attack will be quite different from a normal usage of the system.
- The distance between normal usage patterns will be low, which makes it possible for most of them to group in large clusters (remaining unclassified normal patterns are the false alarms of methods presented in Section 3).

We propose to detect intrusion attempts among the records of a Web server, such as an Apache access log file. For each access on the Web site, such a file

keeps record of: the IP, the date, the requested URL and the referrer (as well as other information, less important in our situation). Our main idea is that the anomalies occuring on two different systems, are highly probable to be attacks. Let us detail the short illustration given in section 1 with A_x, an anomaly that is not an attack on site S_1. A_x is probably a context based anomaly, such as a new kind of usage specific to S_1. Therefore, A_x will not occur on S_2. As an illustration, let us consider a php request on the staff directory for a new employee: John Doe, who works in room 204, floor 2, in the R&D department. The request will have the following form: `staff.php?FName=John\&LName=Doe \&room=204\&floor=2\&Dpt=RD`. This new request, due to the recent recruitment of John Due in this department should not be considered as an attack. However, with an IDS based on outlier detection, it is likely to be considered as an intrusion, since it is not an usual behaviour.

Let us now consider A_y, an anomaly corresponding to a true intrusion. Let us consider that A_y is based on a security hole of the system (for instance a php vulnerability). Then A_y will be the same for every site attacked through this weakness. For instance, a php request corresponding to an attack might look like: `staff.php?path=/../etc/passwd%00`. In this request, one can see that the parameters are not related to the data accessed by the php script, but rather to a security hole that has been discovered on the *staff* script that returns passwords. If this script is provided by a software company to many firms, the usage of this security hole will repeatedly occur on different sites and the request having parameter `path=/../etc/passwd%00` will be the same for all the victims.

For clarity of presentation we present our framework on the collaboration of two Web sites, S_1 and S_2 and we consider the requests that have been received by the scripts of each site (cgi, php, sql, etc). Our goal is to perform a clustering on the usage patterns of each site and to find the common outliers. However, that would not be enough to meet the second constraint of our objective: the request of only one parameter, *n*, the number of alarms to return. Our distance measure (presented in section 4.1) will allow normal usage patterns to be grouped together rather than to be mixed with intrusion patterns. Moreover, our distance measure has to distinguish an intrusion pattern from normal usage patterns and also from other intrusion patterns (since different intrusion patterns will be based on a different security hole and will have very different characteristics). Our algorithm performs successive clustering steps for each site. At each step we check the potentially matching outliers between both sites. The clustering algorithm is agglomerative and depends on the maximum distance (MD) requested between two objects.

Let us consider that *n*, the desired number of alarms, is set to 1 and the usage patterns are distributed as illustrated in figure 1. Let us also consider that, for these sites, cluster *A* at step 1 is the only one corresponding to an intrusion attempt. For the first step, MD is initialized with a very low value, so the clusters will be as tight and small as possible. Then we check correspondences between outliers of S_1 and S_2. Let us consider the clustering results on S_1 and S_2 at step 1 in figure 1. There are two matching outliers between both sites (*A* and *B*). That would lead to 2 alarms (just one of the alarms being true) which represents more than the number of alarms

desired by the user. We thus have to increase the clustering tolerance (*i.e.* increase MD) so that bigger clusters can be built. After a few steps, we will find the clusters of step *n* in figure 1. The only common outlier is *A*, which corresponds to the intrusion attempt. Furthermore, this will trigger one alarm, as desired by the user, and there is no need to continue increasing MD until step *m*.
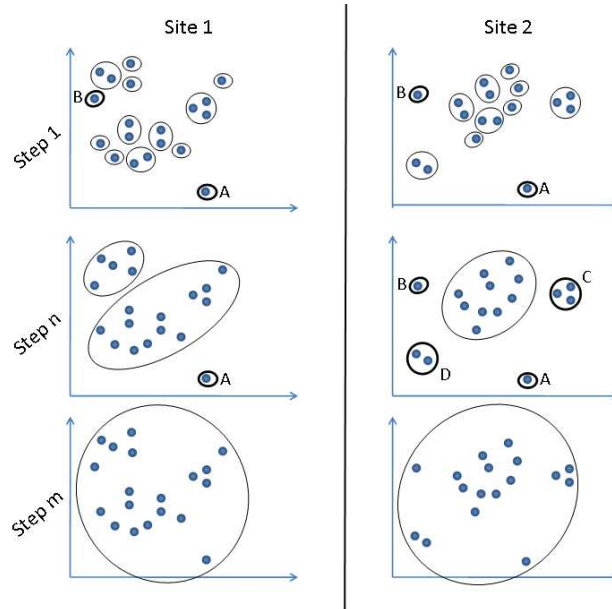
Fig. 1 Detection of common outliers in the usage patterns of two Web sites

As explained in section 1, we want to propose an algorithm that requires only one parameter, *n*, the maximum number of alarms desired by the end-user. Actually, this work is intended to explore the solutions for monitoring a network in real time. Then, the potential alarms will be triggered at each step of the monitoring (for instance with a frequency of one hour). A first batch of usage data is clustered on each site and *n* alarms are triggered. Depending on the number of true or false alarms, the user might want to adjust *n* for the next step, until no false alarm is returned. Our assumption is that common outliers, sorted by similarity from one site to another, will give the intrusions at the beginning of the list.

Obviously, such a framework requires a good privacy management of each partner's data. This is a very important issue in our framework and we propose solutions in chapter **??** (note to the editors, please insert here the chapter number for Verma et. al, in this book).

Our challenge is to reply to important questions underlying our method ; what is the distance between two usage patterns? How to separate clusters in order to give the list of outliers? How to detect common outliers?

Our main algorithm, corresponding to the framework presented in this section, is given in section 4.1. Our distance measure and our clustering algorithm are given in section 4.2. As explained in section 4.3 our outlier detection method is parameterless, thanks to a wavelet transform on the cluster distribution. In contrast to most previous methods [Jin et al., 2001, Zhong et al., 2007, Portnoy et al., 2001, Joshua Oldmeadow et al., 2004] it does not require a percent of cluster and it does not depend on a top-n parameter given by the user. The correspondance between outliers of S_1 and S_2 also has to be parameterless. As explained in section 4.4 it will automatically find clusters that are close enough to trigger an alarm.

3 Related Work

Atypical data discovery is a really active research topic for now few decades. The problem of finding in databases patterns that deviate significantly from a well-defined notion of normality, also called *outlier detection*, has indeed a wide range of applications, such as fraud detection for credit card [Aleskerov et al., 1997], health care [Spence et al., 2001], cyber security [Ertoz et al., 2004] or safety of critical systems [Fujimaki et al., 2005].

Over time many techniques have been developped to detect outliers, leading to a number of surveys and review articles [Hodge and Austin, 2004, Chandola et al., 2008]. Some of them more precisely focus on the topic of outlier detection within the context of intrusion detection in computer networks [Lazarevic et al., 2003, Patcha and Park, 2007]. We focus on this specific area and we propose an *unsupervised* anomaly-based detection system. On the opposite to *semi-supervised* anomaly detection systems, consisting of describing normal behaviours to detect deviating patterns [Marchette, 1999, Wu and Zhang, 2003, Vinueza and Grudic, 2004], *unsupervised* techniques do not require a preliminary identification of the normal usage by a human expert. Our application will thus be more usable in a real-world context.

Statistic community has quite extensively studied the concept of outlyingness [Barnett and T. Lewis, 1994, Markou and Singh, 2003, Kwitt and Hofmann, 2007]. Statistical approaches construct probability distribution models under which outliers are objects of low probability [Rousseeuw and Leroy, 1996, Billor et al., 2000, Lee and Xiang, 2001] However, within the context of intrusion detection, dimensionality of data is high. Therefore, to improve overall performance and accuracy, it has become necessary to develop data mining algorithms using the whole data distribution as well as most of data features [Knorr and Ng, 1998, Breunig et al., 2000, Aggarwal and Yu, 2001].

Most of these approaches are based on clustering-based outlier detection algorithms [Ester et al., 1996, Portnoy et al., 2001, Eskin et al., 2002, He et al., 2003, Papadimitriou et al., 2003]. Such techniques rely on the assumption [Chandola et al., 2008] that normal points belong to large and dense clusters while anomalies (or outliers, atypical instances) either do not belong to any clusters [Knorr and Ng, 1998, Ramaswamy et al., 2000, Duan et al., 2006] or form very small (or very sparse)

clusters [Otey et al., 2003, Chimphlee et al., 2005, Pires and Santos-Pereira, 2005, Fan et al., 2006]. In other words anomaly detection consists in identifying those among the data that are far from significant clusters – either isolated or in small clusters. Depending on the approach, the number of parameters required to run the algorithm can be high and will lead to different outliers. To avoid this, some works return a ranked list of potential outliers and limit the number of parameters to be specified [Ramaswamy et al., 2000, Jin et al., 2001, Fan et al., 2006].

However all the anomaly-based intrusion detection techniques suffer of the number of false alarms they trigger. On the contrary, misuse techniques (*i.e.* approaches that detect elements similar to well-known malicious usage) will precisely detect attacks but they will miss every intrusion that differs from these already known attack signatures. Therefore some works proposed collaborative frameworks in order to improve performance and both true and false alarm rates [Valdes and Skinner, 2001, Locasto et al., 2004, Yegneswaran et al., 2004]. These approaches rely on propagating in a distributed IDS IP blacklist after individual misuse or anomaly detection. Also this communication can lead to more accurate results, it does not allow the system to uncover totally unknown attacks or to avoid high false alarm rates.

For these reasons we propose an anomaly detection approach that uses collaboration between systems in order to discriminate attacks from emerging or novel usage behaviours, thus leading to a reduced number of false alarms. To the best of our knoweldge, this is the first proposal for such an IDS.

4 COD: Common Outlier Detection

The principle of COD is to perform successive clustering on usage patterns of different partners sites, until the number of common outliers meets the number of alarms desired by the user. We present in this section an algorithm designed for two information systems. Extending this work to more than two systems would require a central node coordinating the comparisons and triggering the alarms, or a peer-to-peer communication protocol. This is not the goal of this chapter, since we want to focus on proposing solutions to the following issues:

- Clustering the usage patterns of a Web site with different levels of MD.
- Proposing a distance measure adapted to intrusion detection.
- Identifying the outliers after having clustered the usage patterns.
- Comparing the outliers given by each partner.

Our objects are the parameters given to script files in the requests received on a Web site. In other words, the access log file is filtered and we only keep lines corresponding to requests with parameters to a script. For each such line, we separate the parameters and for each parameter we create an object. Let us consider, for instance, the following request: `staff.php?FName=John&LName=Doe`. The corresponding objects are $o_1 =$ `John` and $o_2 =$ `Doe`. Once the objects are obtained

from the usage data of multiple Web sites, COD is applied and gives their common outliers.

4.1 Main Algorithm

As explained in section 2.2, COD will process the usage patterns of both sites step by step. For each step, a clustering result is provided and analyzed for intrusion detection. The pseudo-code is given in figure 2. First, *MD* is set to obtain very tight and numerous clusters (very short distance is allowed between two objects in a cluster). Then, *MD* is relaxed by an amount of 0.05 step after step in order to increase the size of resulting clusters, decrease their number and lower the number of alarms. When the number of alarms desired by the user is reached, then COD ends.

Algorithm Cod
Input: U_1 and U_2 the usage patterns of sites S_1 and S_2
       and n the number of alarms.
Output: I the set of clusters corresponding
        to malicious patterns.

1. $MD \leftarrow 0$;
2. $MD \leftarrow MD + 0.05$;
3. $C_1 \leftarrow Clustering(U_1, MD)$;
   $C_2 \leftarrow Clustering(U_2, MD)$;
4. $O_1 \leftarrow Outliers(C_1)$; $O_2 \leftarrow Outliers(C_2)$;
5. $I \leftarrow CommonOutliers(O_1, O_2, MD)$;
6. If $|I| \leq n$ then return I ;
7. If $MD = 1$ then return I ; // No common outlier
8. Else return to step 2 ;

End algorithm Cod

Fig. 2 Algorithm Cod

4.2 Clustering

COD clustering algorithm (given in figure 3) is based on an agglomerative principle. The goal is to increase the volume of clusters by adding candidate objects, until the Maximum Distance (MD) is broken (*i.e.* there is one object o_i in the cluster such that the distance between o_i and the candidate object o_c is greater than MD).
Distance between objects. We consider each object as a sequence of characters. Firstly, we need to introduce the notion of subsequence in definition 1.

Definition 1. Let $S = s_1, s_2, \ldots, s_n$ be a sequence of characters having length n, a *subsequence* is a subset of the characters of S with respect to their original order. More formally, $V = v_1, v_2, \ldots, v_k$, having length $k \leq n$, is a subsequence of S if there exist integers $i_1 < i_2 < \ldots < i_k$ such that $s_1 = v_{i_1}$, $s_2 = v_{i_2}$, $\ldots$ $s_k = v_{i_k}$.

Our distance is then based on the longest common subsequence (LCS, where the problem is to find the maximum possible common subsequence of two sequences), as described in definition 2.

Definition 2. Let s_1 and s_2 be two sequences. Let $LCS(s_1, s_2)$ be the length of the longest common subsequences between s_1 and s_2. The *distance $d(s_1, s_2)$* between s_1 and s_2 is defined as follows:

$$d(s_1, s_2) = 1 - \frac{2 \times LCS(s_1, s_2)}{|s_1| + |s_2|}$$

Example 1. Let us consider two parameters p_1=intrusion and p_2=induction. The LCS between p_1 and p_2 is L=inuion. L has length 6 and the dissimilarity between p_1 and p_2 is $d = 1 - \frac{2 \times L}{|p_1| + |p_2|} = 33.33\%$. Which also means a similarity of 66.66% between both parameters.

Centre of clusters. When an object is inserted into a cluster we maintain the centre of this cluster, since it will be used in the CommonOutliers algorithm described in Figure 6. The centre of a cluster C is the LCS between all the objects in C. When object o_i is added to C, its center C_c is updated. The new value of C_c is the LCS between the current value of C_c and o_i.

4.3 Wavelet-based Outlier Detection

Most previous work in outlier detection require a parameter [Jin et al., 2001, Zhong et al., 2007, Portnoy et al., 2001, Joshua Oldmeadow et al., 2004], such as a percent of small clusters that should be considered as outliers, or the top-n outliers. Their key idea is generally to sort the clusters by size and/or tightness. We consider that our clusters will be as tight as possible, according to our clustering algorithm and we want to extract outliers by sorting the cluster by size. The problem is to separate "big" and "small" clusters. Our solution is based on an analysis of cluster distribution, once they are sorted by size. The usual distribution of clusters is illustrated by Figure 4 (screenshot made with our real data). In [Marascu and Masseglia, 2009], the authors proposed to use a wavelet transform to cut down the distribution. This technique is illustrated by figure 4, where the y axis stands for the size of the clusters, whereas their index in the sorted list is represented on x, and the two plateaux allow separating small and big clusters. With a prior knowledge on the number of plateaus (we want two plateaus, the first one standing for small clusters, or outliers, and the second one standing for big clusters) we can cut the distribution in a very effective

Algorithm Clustering

**Input:** U, the usage patterns

and MD, the Maximum Distance.

Output: C, the set of as large clusters as possible,

respecting MD.

1. Build M, the distance matrix between each pattern in U;
2. $\forall p \in M, Neighbours_p \leftarrow$ sorted list of neighbours for p (the first usage pattern in the list of p is the closest to p).
3. $DensityList \leftarrow$ sorted list of patterns by density ;
4. $i \leftarrow 0$; $C \leftarrow \emptyset$;
5. $p \leftarrow$ next unclassified pattern in $DensityList$;
6. $i++$; $c_i \leftarrow p$;
7. $C \leftarrow C + c_i$;
8. $q \leftarrow$ next unclassified pattern in $Neighbours_p$;
9. $\forall o \in c_i$, If $distance(o,q) > MD$ then return to step 5 ;
10. add q to c_i ;
11. $C_c \leftarrow LCS(C_c, q)$; //C_c is the center of C
12. return to step 8 ;
13. If unclassified patterns remain then return to step 5 ;
14. return C ;

End algorithm Clustering

Fig. 3 Algorithm Clustering

manner. Actually, each cluster mapped to the first plateau will be considered as an outlier.
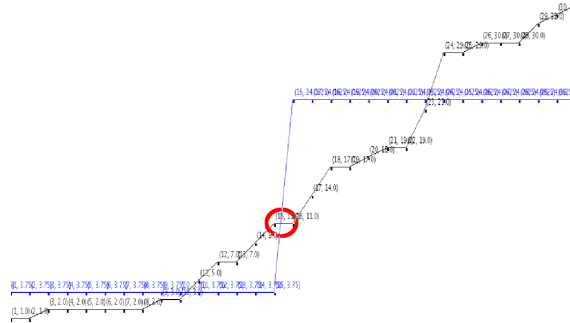
Fig. 4 Detection of outliers by means of Haar Wavelets

Advantages of this method, for our problem, are illustrated in figure 5. Depending on the distribution, wavelets will give different indices (where to cut). For instance, with few clusters having the maximum size (see graph with solid lines from figure 5), wavelets will cut the distribution in the middle. On the other hand, with a large number of such large clusters (see graph with dashed lines from figure 5), wavelets

will accordingly increase the number of clusters in the little plateau (taking into account the large number of big clusters).
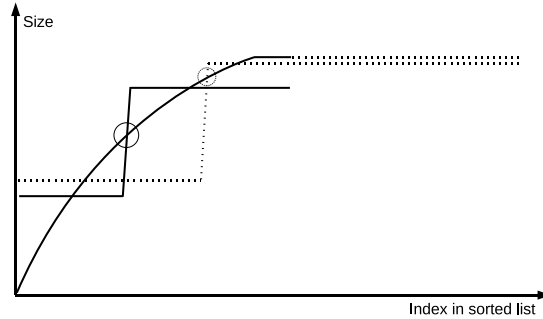
Fig. 5 Self-adjusting detection of outliers

4.4 Comparing Outliers

Since we want our global algorithm to require only one parameter (the number of alarms), we want to avoid introducing a similarity degree for comparing two lists of outliers. Our algorithm (given in figure 6) for this comparison will use the centre of outliers. For each pair of outliers, CommonOutliers calculates the distance between centers of these outliers. If this distance is below the current MD (C.f. Subsection 4.2), then we consider those outliers as similar and add them to the alarm list. The centre of an outlier is the LCS calculated on all the objects in this outlier. The distance between two outliers is given by the LCS between the centers.

5 Experiments

The goal of this section is to analyze our results (*i.e.* the number of outliers and true intrusions and the kind of intrusions we have detected).

5.1 Datasets

Our datasets come from two different research organizations; Inria Sophia-Antipolis and IRISA. We have analyzed their Web access log files from March 1 to March 31. The first log file represents 1.8 Gb of rough data. In this file, the total num-

Algorithm CommonOutliers
Input: O_1 and O_2, two lists of outliers
          and *MD*, the maximum distance.
Output: *A*, the list of alarms (common outliers).

1. $A \leftarrow \emptyset$
2. $\forall i \in O_1$ do
3. $\forall j \in O_2$ do
4. $centre_i \leftarrow centre(i)$;
5. $centre_j \leftarrow centre(j)$;
6.   If $distance(centre_i, centre_j) < MD$
                Then $A \leftarrow A + i \cup j$;
7.   done ;
8. done ;
9. Return A ;

End algorithm CommonOutliers

Fig. 6 Algorithm CommonOutliers

ber of objects (parameters given to scripts) is 30,454. The second log file represents 1.2 Gb of rough data and the total number of objects is 72,381. COD has been written in Java and C++ on a PC (2.33GHz i686) running Linux with 4Gb of main memory. Parameters that are automatically generated by the scripts have been removed from the datasets since they cannot correspond to attacks (for instance "`publications.php?Category=Books`"). This can be done by listing all the possible combinations of parameters in the scripts of a Web site.

5.2 Detection of common outliers

As described in Section 2.2, COD proceeds by steps and slowly increases the value of MD, which stands for a tolerance value when grouping objects during the clustering process. In our experiments, MD has been increased by steps of 0.05 from 0.05 to 0.5. For each step, we report our measures in table 1. The meaning of each measure is as follows. C_1 (resp C_2) is the number of clusters in site 1 (resp. site 2). O_1 (resp. O_2) is the number of outlying objects in site 1 (resp. site 2). $\%_1$ (resp $\%_2$) is the fraction of outlying objects on the number of objects in site 1 (resp. site 2). For instance, when MD is set to 0.3, for site 1 we have 6,940 clusters (built from the 30,454 objects) and 5,607 outlying objects, which represents 18.4% of the total number of objects in site 1. *COD* is the number of common outliers between both sites and $\%_{FP}$ is the percentage of false positive alarms within the common outliers. For instance, when MD is set to 0.05, we find 101 alarms among which 5 are false (which represents 4.9%). One first observation is that outliers cannot be directly used to trigger alarms. Obviously, a number as high as 5,607 alarms to check, even

for one month, is not realistic. On the other hand, the results of COD show its ability to separate malicious behaviour from normal usage.

Our false positive patterns correspond to normal requests that are common to both sites but rarely occur. For instance, on the references interrogation script of Inria Sophia-Antipolis, a user might request papers of "John Doe" and the request will look like `publications.php?FName=John\&LName=Doe`. If another user requests papers of "John Rare" on the Web site of IRISA, the request will be `biblio.php?FName=John\&LName=Rare` and the parameter "John" will be given as a common outlier and trigger an alarm. As we can see, $\%_{FP}$ is very low (usually we have at most 5 false alarms in our experiments for both Web sites) compared to the thousands of outliers that have been filtered by COD.

Another lesson from these experiments is that a low MD implies very small clusters and numerous outliers. These outliers are shared between both sites, among which some are false alarms due to rare but common normal usage. When MD increases, the clustering process gets more agglomerative and alarms are grouped together. Then one alarm can cover several ones of the same kind (*e.g.* the case of easter eggs explained further). At the same time, the number of outliers corresponding to normal usage decreases (since they are also grouped together). Eventually, a too large value of MD implies building clusters that do not really make sense. In this case, outliers will get larger, and the matching criteria will get too tolerant, leading to a large number of matching outliers capturing normal usage.

In a streaming environment, one could decide to keep 70 as the number of desired alarms and watch the ratio of false positive alarms. If this ratio decreases, then the end-user should consider increasing the number of desired alarms.

Measure \MD	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
C_1	14165	11922	10380	8974	7898	6940	6095	5390	4863	4316
O_1	13197	10860	8839	7714	6547	5607	5184	4410	3945	3532
$\%_1$	43.3%	35.6%	29%	25.3%	21.5%	18.4%	17%	14.4%	12.9%	11.6%
C_2	37384	30456	25329	21682	19080	16328	14518	12753	10984	9484
O_2	35983	27519	24032	20948	18152	14664	12738	11680	10179	8734
$\%_2$	49.6%	37.9%	33.1%	28.9%	25%	20.2%	17.5%	16.1%	14%	12.1%
COD	101	78	74	70	67	71	71	85	89	90
$\%_{FP}$	4.9%	5.12%	4%	2.85%	1.5%	2.8%	2.8%	10.6%	11.2%	16.6%

Table 1 Results on real data

5.3 Execution times

Processing one month of real data: We want to show that COD is suitable both for off-line and on-line environments. First, regarding off-line environments, we report in Figure 7 the time responses of COD for the results presented in subsection 5.2.

These results have been obtained on real log files corresponding to the navigations of one month from Inria Sophia-Antipolis and IRISA for a rough size of 1.8 Gb and 1.2 Gb. We consider that preprocessing the log files and obtaining the clusters and outliers for each site can be done separately and by different machines. This is why Figure 7 reports the maximum time for Sophia-Antipolis (Log1) and IRISA (Log2) for these three steps (preprocessing, clustering and outlier detection). Foreach user threshold, we also report the execution time of common outlier detection (COD). Eventually, the total time (addition of preprocessing, clustering, outlier detection and common outlier detection) foreach threshold is reported. The global time for these two log files (corresponding to one month) is 2819 minutes (addition of all the total times).
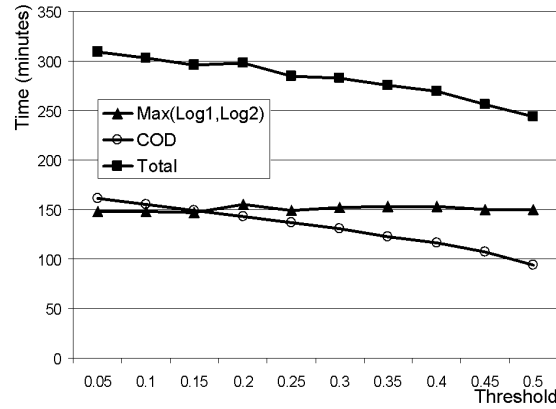

Fig. 7 Execution times of COD for one month of real data

Once this knowledge is obtained (*i.e.* the outliers for each site), when new trans-actions arrive in the system (new navigations on the site, for instance) we want to extract the outliers for this set of new navigations, and compare them to the existing ones. The time responses obtained for this real-time situation are reported hereafter.

Time response for one day: the navigations, after one day, represent approxi-mately 60 Mo of rough data, and 2500 objects in average after preprocessing. Pars-ing one day of data needs 73 seconds in average. Clustering and outlier detection needs 82 seconds. Common outlier detection requests comparing 715 outliers (av-erage number of outliers for one day) to 21,460 known outliers (average number for one month) over 5 thresholds. The total time of common outlier detection for one day of navigations is 43 minutes.

Time response for one hour: with one hour of navigations, the total time for detecting outliers shared with one month of navigations, from the partner site, is less than 2 minutes.

5.4 A sample of our results

None of the attacks found in our experiments have been successful on the considered Web sites. However, our security services and our own investigations allow us to confirm the intrusion attempts that have been discovered by our method:

- **Code Injection:** a recent kind of attack aims to inject code in PHP scripts by giving a URL in the parameters. Here is a sample of such URLs detected by COD:

  - http://myweddingphotos.by.ru/images?
  - http://levispotparty.eclub.lv/images?
  - http://0xg3458.hub.io/pb.php?

  Depending on the PHP settings on the victim's Web server, the injected code allows modifying the site. These URLs are directly, automatically and massively given as parameters to scripts through batches of instructions.
- **Passwords:** another kind of (naive and basic) attack aims to retrieve the password file. This results in outliers containing parameters like `../etc/password` with a varying number of `../` at the beginning of the parameter. This is probably the most frequent attempt. It is generally not dangerous but shows the effectiveness of our method.
- **Easter Eggs:** this is not really an intrusion but if one adds the code ?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 to the end of any URL that is a PHP page, he will see a (funny) picture on most servers. Also on April 1st (April Fool's Day), the picture will replace the PHP logo on any phpinfo() page. This code (as well as two other ones, grouped into the same outlier) has been detected as a common outlier by COD.

6 Conclusion

We have proposed i) an unsupervised clustering scheme for isolating atypical behaviours, ii) a parameterless outlier detection method based on wavelets and iii) a new feature for characterizing intrusions. This new feature is based on the repetition of an intrusion attempt from one system to another. Actually, our experiments show that atypical behaviours (up to several thousands for one day on Inria Sophia-Antipolis) cannot be directly used to trigger alarms since most of them correspond to normal (though atypical) requests. On the other hand, this very large number of outliers can be effectively filtered in order to find true intrusion attempts (or attacks) if we consider more than one site. By comparing the outliers of two sites, our method kept only less than one hundred alarms, reducing the amount of atypical behaviours up to 0.21%. Eventually, our method guarantees a very low ratio of false alarms, thus making unsupervised clustering for intrusion detection effective, realistic and feasible.

Acknowledgement

The authors want to thank Laurent Mirtain, the responsible for intrusion detection of Inria Sophia-Antipolis, for his assistance in identifying attacks in our access log files.

References

[Aggarwal and Yu, 2001] Aggarwal, C. C. and Yu, P. S. (2001). Outlier detection for high dimensional data. *SIGMOD Records*, 30(2):37–46.

[Aleskerov et al., 1997] Aleskerov, E., Freisleben, B., and Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. In *IEEE Computational Intelligence for Financial Engineering*.

[Barbara et al., 2001] Barbara, D., Wu, N., and Jajodia, S. (2001). Detecting novel network intrusions using bayes estimators. In *1st SIAM Conference on Data Mining*.

[Barnett and T. Lewis, 1994] Barnett, V. and T. Lewis, T., editors (1994). *Outliers in statistical data*. John Wiley & Sons.

[Billor et al., 2000] Billor, N., Hadi, A. S., and Velleman, P. F. (2000). BACON: blocked adaptive computationally efficient outlier nominators. *Computational Statistics and Data Analysis*, 34.

[Bloedorn et al., 2001] Bloedorn, E., Christiansen, A. D., Hill, W., Skorupka, C., and Talbot, L. M. (2001). Data mining for network intrusion detection: How to get started. Technical report, MITRE.

[Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. *SIGMOD Records*, 29(2):93–104.

[Chandola et al., 2008] Chandola, V., Banerjee, A., and Kumar, V. (2008). Anomaly detection - a survey. *ACM Computing Surveys*.

[Chimphlee et al., 2005] Chimphlee, W., Abdullah, A. H., Md Sap, M. N., and Chimphlee, S. (2005). Unsupervised anomaly detection with unlabeled data using clustering. In *International conference on information and communication technology*.

[Dokas et al., 2002] Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., and Tan, P. (2002). Data mining for network intrusion detection. In *NSF Workshop on Next Generation Data Mining*.

[Duan et al., 2006] Duan, L., Xiong, D., Lee, J., and Guo, F. (2006). A local density based spatial clustering algorithm with noise. In *IEEE International Conference on Systems, Man and Cybernetics*.

[Ertoz et al., 2004] Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.-N., Kumar, V., Srivastava, J., and Dokas, P. (2004). Minds - minnesota intrusion detection system. *Data Mining - Next Generation Challenges and Future Directions*.

[Eskin et al., 2002] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of Data Mining in Computer Security*.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density–based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining*.

[Fan et al., 2006] Fan, H., Zaiane, O. R., Foss, A., and Wu, J. (2006). A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. In *Pacific-Asia conference on knowledge discovery and data mining*.

[Fujimaki et al., 2005] Fujimaki, R., Yairi, T., and Machida, K. (2005). An approach to spacecraft anomaly detection problem using kernel feature space. In *11th ACM SIGKDD international conference on Knowledge discovery in data mining*.

[He et al., 2003] He, Z., Xu, X., and Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24.

[Hodge and Austin, 2004] Hodge, V. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22.

[Jin et al., 2001] Jin, W., Tung, A. K. H., and Han, J. (2001). Mining top-n local outliers in large databases. In *7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 293–298.

[Joshua Oldmeadow et al., 2004] Joshua Oldmeadow, J., Ravinutala, S., and Leckie, C. (2004). Adaptive clustering for network intrusion detection. In *8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 255–259.

[Knorr and Ng, 1998] Knorr, E. M. and Ng, R. T. (1998). Algorithms for mining distance-based outliers in large datasets. In *24rd International Conference on Very Large Data Bases*, pages 392–403.

[Kwitt and Hofmann, 2007] Kwitt, R. and Hofmann, U. (2007). Unsupervised anomaly detection in network traffic by means of robust pca. In *International Multi-Conference on Computing in the Global Information Technology*.

[Lazarevic et al., 2003] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., and Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *3rd SIAM International Conference on Data Mining*.

[Lee and Stolfo, 1998] Lee, W. and Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In *7th conference on USENIX Security Symposium*.

[Lee and Xiang, 2001] Lee, W. and Xiang, D. (2001). Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*.

[Locasto et al., 2004] Locasto, M., Parekh, J., Stolfo, S., Keromytis, A., Malkin, T., and Misra, V. (2004). Collaborative distributed intrusion detection. Technical Report CUCS-012-04, Columbia Unviersity Technical Report.

[Marascu and Masseglia, 2009] Marascu, A. and Masseglia, F. (2009). Parameterless outlier detection in data streams. In *SAC*, pages 1491–1495.

[Marchette, 1999] Marchette, D. (1999). A statistical method for profiling network traffic. In *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 119–128.

[Markou and Singh, 2003] Markou, M. and Singh, S. (2003). Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, 83.

[Otey et al., 2003] Otey, M., Parthasarathy, S., Ghoting, A., Li, G., Narravula, S., and Panda, D. (2003). Towards nic–based intrusion detection. In *9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–728.

[Papadimitriou et al., 2003] Papadimitriou, S., Kitagawa, H., Gibbons, P., and Faloutsos, C. (2003). LOCI: fast outlier detection using the local correlation integral. In *19th International Conference on Data Engineering*.

[Patcha and Park, 2007] Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Networks*, 51.

[Pires and Santos-Pereira, 2005] Pires, A. and Santos-Pereira, C. (2005). Using clustering and robust estimators to detect outliers in multivariate data. In *International Conference on Robust Statistics*.

[Portnoy et al., 2001] Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*.

[Ramaswamy et al., 2000] Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *SIGMOD Records*, 29(2):427–438.

[Roesch, 1998] Roesch, M. (1998). SNORT.

[Rousseeuw and Leroy, 1996] Rousseeuw, P. and Leroy, A. M., editors (1996). *Robust Regression and Outlier Detection*. Wiley-IEEE.

[Spence et al., 2001] Spence, C., Parra, L., and Sajda, P. (2001). Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*.

[Valdes and Skinner, 2001] Valdes, A. and Skinner, K. (2001). Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*, pages 54–68.

[Vinueza and Grudic, 2004] Vinueza, A. and Grudic, G. (2004). Unsupervised outlier detection and semi–supervised learning. Technical Report CU-CS-976-04, Univ. of Colorado at Boulder.

[Wu and Zhang, 2003] Wu, N. and Zhang, J. (2003). Factor analysis based anomaly detection. In *IEEE Workshop on Information Assurance*.

[Yegneswaran et al., 2004] Yegneswaran, V., Barford, P., and Jha, S. (2004). Global intrusion detection in the domino overlay system. In *Network and Distributed Security Symposium*.

[Zhong et al., 2007] Zhong, S., Khoshgoftaar, T. M., and Seliya, N. (2007). Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, 14.