Triple-Similarity Mechanism for Alarm Management in the Cloud

Bruno L. Dalmazo[1], João P. Vilela, Marilia Curado

{*dalmazo, jpvilela, marilia*}*@dei.uc.pt*
CISUC, Department of Informatics Engineering
University of Coimbra, Coimbra, Portugal

Abstract

Its distributed nature and ubiquitous service make the cloud subject to several vulnerabilities. One of the main tools used for reporting suspicious activity in the network's traffic is the Intrusion Detection System. However, two significant problems arise: the huge volume of control messages between the virtual machines and the servers; and the associated transfer costs. In this work, we propose a Triple-Similarity Mechanism (T-SyM) for grouping similar alarms that may correspond to the same attack (or attempt) in order to reduce the number of messages and, consequently, the total amount of information. In addition, we propose an algorithm for calculating the severity level of the alarms. T-SyM works on the basis of 3 steps: individual similarity (Euclidian distance), clustering relevant features (k-means algorithm) and generating the output (the Tanimoto coefficient). An evaluation of the most common attacks is performed using real traces from an IDS. Our mechanism was able to decrease the number of alarms by up to 90% and reduce the total amount of data by more than 80%.

Key words: Alarm management; similarity analysis; security; network traffic; cloud computing.

1. Introduction

An Intrusion Detection System (IDS) is designed to monitor a system or a network in order to report any suspicious activity that may compromise its operation. The report of the suspect activity represents an output of the IDS, namely, an alarm. Usually, alarms carry information about the suspicious activity such as: type of attack, the timestamp, the number of packets, the IP address and the port number. Thus, alarms are considered valuable information to support the administrator in decision-making about whether it is a true attack or a false alarm which came from one or more collaborative IDSs [1] [2].

[1]Corresponding author

An IDS may be based on two main approaches to recognize an attack (or attempt) that differ in the way the data is analysed and processed. The signature approach refers to the detection of attacks by looking for specific patterns based on other similar attacks, while the anomaly approach consists in searching for deviations from proper behaviour through periodic observations of the system. Signature-based detection methods usually present a low number of false alarms but do not have the ability to detect new or variants of known attacks, while anomaly-based detection has the benefit that a new attack, for which a signature does not exist, can be detected if it falls out of the regular traffic patterns.

Intrusion detection in a cloud environment involves other aspects that need to be considered, for instance, the relationship between the server and the virtual machine (VM). Usually, a server may host hundreds of virtual machines that provide different services, for instance, storage, web server, e-mail, and others [3]. Another important feature relates to where the information in question will be collected and processed. In this case, the information may come from the infrastructure, platforms of software development or applications. Furthermore, the distributed architecture design of clouds is seen as the key point on which IDSs rely for detecting threats.

The distributed nature and ubiquitous service make cloud computing vulnerable to several types of attacks. For example: a denial of service attack, data privacy and integrity, identity management and access control, and others [4, 5]. Furthermore, the amount of alarms generated can be overwhelming [6], thus requiring alarm management solutions for an effective management of resources. Managing alarms triggered by traditional intrusion detection methods is even more challenging in the cloud computing environment. In this case, the network traffic is apt to undergo sudden changes and these may be easily confused with traffic anomalies [7].

1.1. Open Issues and Requirements for Managing Alarms in the Cloud

In recent years, new approaches regarding alarm management have been proposed in the literature such as alarm correlation [8], regular expression matching [9] and clustering alarms [10]. However, these studies are concerned with increasing the number of true alarms and they fail to respond appropriately to a low number of false alarms or decrease the number of control messages in general [11].

The number of alarms generated over time is even greater in cloud computing. Besides the sudden changes that the traffic suffers due to the elastic and scalable nature of cloud environments, the number of messages increases proportionally with the number of virtual machines. Moreover, it is known that around 99% of the alarms are false both in cloud computing [11] and in traditional environments [12] [13] [14]. The wide disparity between the true and false alarms generated has certainly compromised the performance of IDS.

The problem is further aggravated in cloud computing due to the huge volume of control messages between the virtual machines and the server. This situation makes the detection system inefficient because it provides an unmanageable amount of alarms for the administrators [12]. In addition, according to

a technical report by the University of California, the cost of the data transfer lies in the region of $100 to $150 per terabyte [15]. Therefore, besides reducing the number of alarms and supporting the management of the cloud infrastructure, managing alarms also facilitates minimizing the network's bandwidth and the associated transfer costs.

From these observations a set of key requirements for managing alarms in the cloud emerge, which are listed as follows:

1. **Self-adaptive**: this requirement refers to the model's ability to learn or train itself based on current information, which is different from static approaches. Cloud computing ensures elasticity which provides scalability. In this context, the cloud provider should be able to preserve its operation under conditions of unexpected change by constantly evaluating its own behaviour.
2. **Low message overhead**: as important as decreasing the false alarm rate, a key point is to reduce the number of control messages between the server and virtual machines. An alarm reduction technique is an absolute necessity for solving this problem [13]. This requirement calls for a compatible model for detecting attacks and classifying alarms dynamically, generating the minimum possible workload.
3. **Collaborative**: this requirement is characterized by the sharing and construction of knowledge among multiple information sources in order to accomplish a task. A large number of heterogeneous entities usually have different information, and combing them can potentially provide better alarm management to the cloud networks and their applications. A collaborative approach is particularly well suited to the cloud environment because these entities have to communicate continuously in order to support decision-making.
4. **Distributed**: an approach in which components are located in different virtual machines and coordinate their actions by passing messages represents, in this context, a distributed alarm management. This feature ensures that the VMs interact with each other in order to join forces to recognize an attack. Moreover, adverse events generated by an individual failure may be minimized [16].

By following this set of key requirements, it is possible to devise an alarm management system suitable for cloud computing.

1.2. Contributions and Outline

In order to address these issues and requirements, we have made several contributions in this work: (*i*) grouping similar alarms that may correspond to the same attack or attack attempt in order to reduce the number of messages sent to the server/administrator and; (*ii*) using the number of occurrences of these groups to adjust the severity of a single alarm based on a similarity analysis. From these contributions, we intend to optimize the efficiency for generating

alarms, decreasing the network data traffic to manage IDS and its associated transfer costs.

The remainder of the paper is organized as follows. Section 2 covers some of the most prominent related work. Section 3 provides theoretical basis for the proposal and the rest of the paper. Section 4 describes the proposed solution and the methodology used for this paper, whilst Section 5 presents the evaluation and discusses the results. Section 6 concludes with some final remarks and prospective directions for future research.

2. Related Work

Improving alarm management in the cloud is a useful means of supporting the cloud provider to manage its assets. Besides decreasing the number of false alarms, it may reduce the amount of alerts that need to be handled. In this section, the latest research findings are organized into two parts. First, several approaches for alarm correlation are presented, outlining the main benefits and drawbacks of each technique. Finally, a discussion about the state-of-the-art and the open issues is provided.

2.1. Managing Alarms

There are numerous approaches to improve the quality of alarms by increasing the number of true positives, such as fuzzy logic [11], artificial neural networks [17], decision tree classifier [18], among others. The aim of other approaches to alarm management is to decrease the rate of false alarms by recognizing the relationships between them [13]. The main approaches found in the literature are described below.

Zhichun Li *et al.* [9] applied signature detection techniques for enhancing an IDS looking for vulnerabilities in a high speed network. An approach that uses semantic information can potentially reduce the number of false alarms. They decreased the false alarm rate based on a signature parsing and regular expression matching engine by using the Single PDU Multiple Signature Matching algorithm. However, there is no a real implementation of this proposal yet, just a small prototype implementation which can handle a limited number of protocols.

Parikh and Chen [19] used statistical pattern recognition techniques among multiple sources of information to decrease the cost of operations associated to intrusion detection activities. Different errors in classifying network traffic generate different costs associated with them, for instance, the cost of a false alarm and the useless traffic generated by it. Although the cost minimization strategy has been successful based on a objective function minimization, the capabilities of learning incrementally and adaptively are not assessed which is important due to the dynamically changing characteristic of network traffic in cloud environments.

Lo *et al.* [10] extended the Snort IDS. In this proposal, four modules are created to work in cooperative mode: intrusion detection, alarms clustering, threshold computation and comparison. The intrusion detection is based on analysis

Table 1: Characteristics of the related works concerning requirements for alarm management

Proposal	Self-adaptive	Low message overhead	Collaborative	Distributed	Cloud scenario
Zhichun Li *et al.* [9]	×	×	×	×	×
Parikh and Chen [19]	×	×	√	×	×
Lo *et al.* [10]	×	×	√	√	√
Salem Benferhat *et al.* [8]	√	×	√	√	×
Leau Beng *et al.* [20]	√	√	√	×	×
Elshoush and Osman [21]	×	√	√	√	×

of the number of packets over time. In comparison with pure Snort-based IDS, they showed that the solution spends almost the same time to compute the detection. The benefit of the proposed modules is preventing the system from a single point of failure attack for cloud environment.

Salem Benferhat *et al.* [8] proposed an approach that can be applied for any classifier (e.g. Hidden Naive Bayes, decision tree classifiers, etc) for alarm correlation. The term "expert knowledge" refers to a person who has extensive skill or knowledge in a particular field. Then, they leverage the expert knowledge for increasing the accuracy of the classification model. However, approaches based on Neural Networks have a serious limitation: they learn the training patterns but lose the ability to make generalizations. For instance, a new adaptation of a known attack may not be detected, thus limiting finding slight variations of instances already classified by the model.

Leau Beng *et al.* [20] perform an extensive study about existing efforts to address the identification of similarities and causality relationships between alerts. They elect the main problem that researchers are trying to solve, namely, the generation of large number of alerts and false positives. Furthermore, they point out the most popular alert correlation approaches with their advantages and drawbacks. This paper surveys existing works and does not propose a new solution.

Elshoush and Osman [21] presented a multiple components approach to deal with different features of alert correlation, each responsible for a different aspect of the overall correlation aim. However the sequence order of acting components affects the process performance. In this context, they introduce a method based on the Alert Fusion Algorithm to merge unrelated alerts and thus reducing the number of messages. Nevertheless, the total time needed for processing this approach increases depending on the number of alerts triggered in each component.

2.2. Discussion

This section compares several approaches by taking into account the requirements for managing alarms in a cloud computing environment. In particular, it summarizes the related work regarding key requirements, application in the cloud context and the remaining open issues as illustrated in Table 1.

IDSs are designed to report basic events that are malicious in nature but they are powerless to recognise causal relationships between the consecutive instances of an attack. Approaches focused on alarm management cannot deal with a low

number of messages and a self-adaptive solution in a distributed architecture at the same time. Moreover, the approaches found in related work are not designed to adapt themselves to the elastic and scalable nature of cloud environments because they fail to learn based on current information. Furthermore, the cloud environment lacks an approach for alarm management able to cope with large amounts of information and control messages. In addition, all types of alarms are processed in the same way in the current approaches, without any level of distinction between the severity of the alarms.

In order to tackle these limitations, we introduce a conceptual solution for managing alarms in the cloud network context by means of a similarity analysis between the features extracted from the IDS, in the following section. From this, it is expected to improve the performance in terms of the usage of the network's bandwidth and the number of alarms. More specifically: the solution proposed reduces the network data traffic to manage IDS and its associated transfer costs; and raises the severity of an individual alarm based on the number of occurrences of similar alarms.

3. Background

Our solution focuses on providing a systematic approach to aggregate similar alarms in the context of the cloud network traffic. Generally, cloud computing involves information collected from several sources, for instance, infrastructures, platforms for software development or applications. In addition, in order to provide a theoretical basis for the solution and the rest of the paper, some concepts must be formalized. Therefore, some important definitions are introduced here.

3.1. Anomaly

A network anomaly corresponds to a circumstance in which the network behaviour deviates from its normal operational pattern. The identification of an anomaly requires the network manager to learn beforehand about the nature of normal traffic operation. Traditionally, this requires monitoring the network for a long period of time to extract the characteristics of the normal behaviour of the network. This type of methodology is known in the literature as building a traffic baseline. This procedure is necessary to ensure an adequate degree of protection and this is an important point to make the system more resilient and protect the network's traffic against threats. Once this step has been completed, any type of misuse that is not seen as normal network operation, will be considered an anomaly.

It is important to emphasize that an anomaly in network traffic does not always correspond to the occurrence of an attempt to attack [22]. For instance, an anomaly could be generated by a network failure event or temporary misconfiguration that results in a problem or outage. Usually it occurs soon after installing some new equipment in the infrastructure or updating the system. Network traffic anomalies are also common whenever a new software release becomes available over a relatively short period of time (flash crowd) [23]. Summing up, in the set of existing types of attacks, there is a subset of them that

can generate anomalies in the network traffic, such as: Distributed Denial of Service (DDoS), Sniffer Attack, Remote to Local (R2L), User to Root (U2R), Probe, etc.

3.2. Features

In this work, we consider features as the means to delineate the main characteristics of the attacks and anomalies in the cloud network traffic. Several studies in the literature propose a set of features to describe the network traffic and anomalies. Kind *et al.* [24] selected a subset of 8 relevant features for network anomaly detection. For instance, histograms of source IP addresses and features extracted from the packet header. Moustafa and Slay [25] present nine categories of attacks including patterns of normal traffic. Moreover, they show 49 features that comprise the flow based between the hosts and the inspection of network packets in order to discriminate between the normal or abnormal observations.

In a past work, Bruno Dalmazo *et al.* [26] present an IDS that identifies a set of features to characterise network anomalies for the purpose of their detection. In this context, the set of features considered for the intrusion detection system consists of: the type of the protocol, the port number, IP address, the packet size, the time interval, the number of packets, the variance between real network traffic and predicted network traffic (Δ-variation) and the attack type.

3.3. Alarm

We consider an alarm as a signal for attention. In other words, a warning notice resulting from the perception of an imminent danger. Usually, an alarm is triggered based on anomalies or signatures present in the network. Moreover, it is created taking into account features which characterize an attack (or attempt). The purpose of an alarm annunciation is to alert the cloud operator about deviations from the normal operating conditions, *e.g.* flooding the bandwidth of a single machine or service. The goal of triggering alarms is to prevent, or at least minimise, problems coming from misconfiguration, internal or external threats to the system.

3.4. Similarity

Similarity analysis is a technique which allows us to assess whether certain alarms are considered similar or dissimilar according to the features that describe them. Basically, there are two properties related to the similarity measures: the level and the commutativity. The similarity level relates to how much an entity X is similar to the entity Y, it ranges from 0 to 1. When two entities are identical the similarity level scores the minimum value, zero. Commutativity determines that the similarity level between X and Y is equal to the similarity level between Y and X.

Assessing similarity between features is a central issue in many research areas such as face recognition [27], linguistics [28] and management systems [29].

The importance of finding suitable measures of similarity cannot be overemphasized [29]. The choice of such measures of similarity depends on the type of measurement or representation of the features. The distance between two entities can be used to indicate how similar they are and it can also be defined by the number of operations to convert an entity X in an entity Y.

There are several approaches to measure similarity, for instance: the Manhattan distance, the Chebyshev and Euclidian distance [30]. The Manhattan distance depends on the rotation of the coordinate system but does not depend on its reflection about a coordinate axis or its translation. When Chebyshev is applied in one dimension, the distance is just the absolute value of the differences. If Chebyshev is applied for two dimensions, the distance is equivalent to the planar Manhattan distance. The Euclidean distance measures the similarity between two points in a Euclidean space as illustrated in Equation 1.

$$Sim_{ED}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

The Tanimoto coefficient is another metric used to measure the level of similarity in finite sample sets based on the size of the intersection divided by the size of the union of the sample sets, as illustrated in Equation 2. The Tanimoto coefficient is a variation of the Jaccard index, the difference being that Tanimoto can be applied to non-binary or quantitative data (groups) while Jaccard can only be applied to binary values [31].

$$Sim_{TM}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i} \tag{2}$$

These measures, as well as other concepts and definitions presented in this subsection, are used as the basis for describing the similarity mechanism in the next section.

4. Triple-Similarity Mechanism

The purpose of this proposal is to provide an efficient method to aggregate similar alarms in cloud-based network traffic. Figure 1 depicts the basis of our proposal, by highlighting the application scenario and the main conceptual components.

Monitoring data from cloud infrastructures is the input of the system. From this data, an *Intrusion Detection System* works by supervising any suspicious network activity. Whenever necessary, it triggers alarms that hold vital information about the abnormal activities. At this point, the *Triple-Similarity Mechanism* relies on data from the *Alarms Database* for aggregating alarms and, consequently, reduces the network data traffic and decreases the associated transfer costs. Then, the *Alarm Generator* produces a single alarm at a higher level of severity. We now describe each component in more detail.

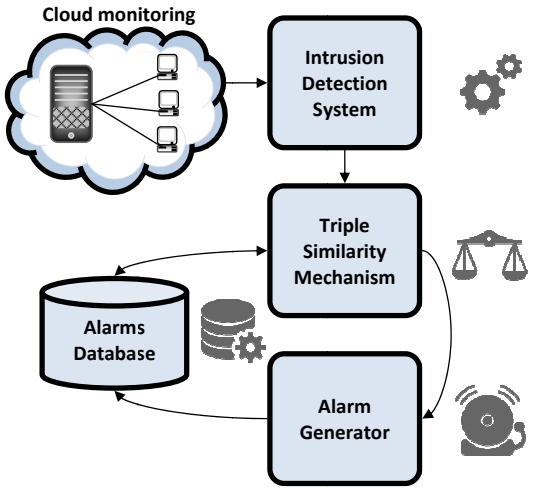


Figure 1: Conceptual components for managing alarms in the cloud

4.1. Intrusion Detection System

At this point, the *Intrusion Detection System* monitors the network traffic generated by the service provided by the virtual machine and its applications during a given period. In this process, raw data is gathered continuously from the cloud network traffic in order to build a network baseline. As an output of this process, the IDS prepares the data collected by measuring the number of packets in the network traffic at regularly spaced intervals, and thus forms a discrete time series ordered by time.

It is important to mention that the similarity mechanism is an open model that could be applied individually or collaboratively for different IDSs. However, for this work we use an IDS that generates alarms based on a distributed mechanism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor [32]. The features are classified into two types: frequency (f) features (*e.g.* the number of times a packet from a protocol appears) and cumulative (c) features (*e.g.* the total number of packets received in a time period), as presented in Table 2.

4.2. Proposed Mechanism

The *Triple-Similarity Mechanism* (T-SyM) acts upon two sets of features: the dataset of alarms and the network traffic. The entire process for determining the similarity level comprises 3 steps: individual similarity, clustering relevant features and generating the output.

- Firstly, the T-SyM needs to measure the individual similarity of each feature with the *Alarms Database*. Giving preference to simplicity, the individual similarity is given by the Euclidean Distance between both features from the alarm dataset and the IDS output. Figure 2 illustrates

Table 2: Details of the features from the IDS

Feature	**Description**
Protocol type (f)	Dividing the network traffic by protocol type facilitates identifying anomalies not visible in the global network traffic
Destination Port number (f) Source Port number (f)	Port number analysis is useful for revealing attacks that attempt to scan ports
Source IP address (f) Destination IP address (f)	This information is useful for recognizing Denial of Service attacks
Packet size (f)	The sudden increase of this feature can indicate a SYN flood attack
Number of packets (c)	Consists of control information and user data used in the prediction
Time interval (c)	A time set containing information between the beginning and the end of the anomaly
Δ-variation (c)	The absolute difference between the real network traffic and the predicted network traffic
Attack type (f)	It describes which is the attack that is being executed
Alarm (f)	Boolean variable that indicates the presence of an alarm

First Similarity which compares two vectors: the vector from the IDS output and the vector from the *Alarms Database* which holds information about previous attacks. These vectors hold all the features that describe the alarm. In order to consider the different types of features generated by the IDS, the similarity approach relies on the set of features responsible for identifying the attack.

- It is then necessary to cluster relevant features based on the individual similarity level, as illustrated at the *Second Similarity* in Figure 1. The feature's behaviour is correlated according to the type of attack. For instance, in a Denial of Service, the attacker uses more than one unique source IP address, often thousands of them, so a high value for this feature is expected (source IP address) [33]. Otherwise, the destination IP address is restricted to only one or a small set, in this case a low number of occurrences for this feature is expected. Therefore, the set of features representing this attack presents a strong individual similarity level for some features and a weak individual similarity level for others. In order to avoid discrepancies between features, a clustering approach is required, namely, the k-means algorithm. This procedure divides the features into two groups: relevant and not relevant.

- Finally, an approach for comparing the similarity of the two groups is needed. The literature presents several techniques with this aim: the Sorensen index, the Jaccard index and the Tanimoto coefficient. However, the Sorensen index and the Jaccard index are metrics that only measure the similarity between objects of purely binary attributes. The Tanimoto coefficient, on the other hand, is not restricted to working with only binary attributes. In *Third Similarity* all similar alarms that correspond to the same attack are aggregated based on the Tanimoto coefficient. At this

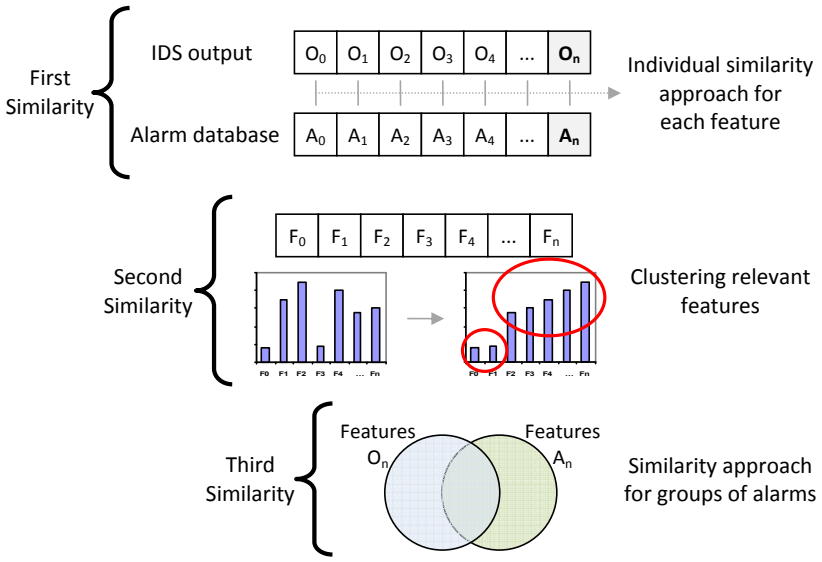


Figure 2: The Triple-Similarity Mechanism

point, this procedure is useful for minimizing the number of control messages sent to the server/administrator of the cloud provider. In addition, the severity of a single alarm increases according to the number of similar occurrences. This procedure is expected to enhance the alarm management in the cloud by reducing the network data traffic and its associated transfer costs.

4.3. Alarms Database

The component called *Alarms Database* is a repository that stores a set of information used to describe an alarm triggered by the IDS. Besides keeping track of the IDS operating history, the *Alarms Database* also includes a finite number of categories by which each alarm is classified. Then, before sending an alarm to the server, the *Triple-Similarity Mechanism* looks for a similar alarm category inside the *Alarms Database*. This is useful to decrease the number of messages, as various alarms can be replaced by just one but with a greater impact.

4.4. Alarm Generator

The IDS causes the system to generate a signal regarding the suspicious activity as soon as it is recognized. Algorithm 1 describes the procedure to calculate the severity level for alarms. The *Alarm Generator* calculates the level of severity based on the *Alarms Database*. Optionally, an *operator* of the cloud infrastructure may interact with the *Alarm Generator* to influence the process for evaluating the level of severity for the alarms. He/she may determine, based on his/her knowledge, how important an attack is, in comparison with others

by increasing the initial severity level. For instance, an operator may configure the algorithm to prioritize an occurrence of DoS attack instead of Portsweep (see variable opK).

Algorithm 1 starts by analysing the network traffic inside a sliding window provided by the IDS. This procedure seeks alarms and classifies them according to the *Alarms Database* and then assigns a level of severity. The ω function calculates the severity based on the number of alarms and the operator's knowledge. The ω function is based on a regression model that is built from the network traffic behaviour [34]. The *Alarm Generator* presents an interval tag that specifies a period of time to wait before sending similar alarms. All information about the attacks and the size of the interval to be monitored is given by the IDS. This is an important aspect that ensures the generalization of the *Triple-Similarity Mechanism* for working with other IDSs.

Algorithm 1 Severity level for alarms

Input: Network traffic, $nTraffic$
Operator knowledge, $opK[\,]$
Output: Severity for alarms, $severity[\,]$

```
1:  Start
2:      procedure GetSeverity(nTraffic, opK[ ])
3:          var slidingWindow = idsWindow(nTraffic)
4:          for each element e in slidingWindow do
5:              if (hasAlarm(e)) then
6:                  var vType = classify(e)
7:                  var nAlarms[vType] + +
8:              end if
9:          end for
10:         for each element e in nAlarms[ ] do
11:             var severity[e] = ω(nAlarms[e], opK[e])
12:         end for
13:         return severity[ ]
14:     end procedure
15: End
```

Without this control mechanism, events of this nature may occur often and cause many alarms to be generated. Moreover, they may correspond recurrently to the same anomaly or attack. To avoid this, alarms will only be sent for the first event and after that, each occurrence with the same features represents an increase in the severity according to the severity level defined by the operator for different types of attacks. This prevents the network and the server from being flooded with redundant alarms as we will now demonstrate in the evaluation.

5. Evaluation and Discussion

To evaluate the effectiveness of the solution, we conducted a case study in which an Intrusion Detection System generates alarms based on a distributed

mechanism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor, as described in [26]. We have considered the DARPA dataset as the case study for evaluation.

5.1. Experimental Environment Setup

The experiments were conducted on a 64-bit standard personal computer with Intel Quad Core i5 Processor with 8Gb of RAM running at 2.70GHz. The operating system was Ubuntu 16.04 LTS. Furthermore, the programming language used for implementing the Triple-Similarity Mechanism was C++ (gcc 4.8.4 c11 version) and several shell scripts to deal with the data.

5.2. DARPA Dataset

The Cyber Security and Information Sciences Group of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory sponsorship, collected the first standard dataset for evaluation of computer network intrusion detection systems [35]. This dataset was the first formal, repeatable, and statistically significant evaluation of intrusion detection systems. We would like to point out that the DARPA data set is a renowned data set for anomaly detection. Although the data set was created in 1998/1999, it is still being used by many works, including recent works in the context of the cloud [36, 37, 38].

The datasets contain data collected from February 1998 up to October 1999. The data consists of three weeks of training data and two weeks of test data. The first and third weeks of the training data do not contain any attack. The second and the fourth week of the training data contains a select subset of labelled attacks. The goal of this work is not evaluate the efficiency of the IDS but decreasing the amount of alarms generated by it. In light of this, we use the second week for the *training phase* because it contains details about the attacks such as timestamp, duration and IP address. In order to make the dataset more realistic, we organized many of the attacks so that the resulting data set consisted of 10% attacks and 90% normal traffic.

The following list describes some attacks included in training data that have been posted on the Lincoln Laboratory web site. All the selected attacks are apt to generate anomalies in the network traffic.

- **Guest**: Try to guess a password via telnet for a guest account (brute force attack)
- **Portsweep**: Surveillance sweep through many ports to determine which services are supported on a single host
- **Ipsweep**: Surveillance sweep performing either a port sweep or ping on multiple host addresses
- **Land**: Denial of Service where a remote host is sent a UDP packet with the same source and destination

- **Back**: Denial of Service attack against an apache webserver where a client requests a URL containing many backslashes
- **Syslog**: Denial of Service for the syslog service connects to port 514 with unresolvable source IP
- **Teardrop**: Denial of service where mis-fragmented UDP packets force a server to reboot

Currently, many providers have been suffering from Distributed DoS attacks, which, for instance, can cause many alarms to be generated [39]. The next section presents the results from applying T-SyM in order to reduce the number of alarms generated.

5.3. Results

Table 3 shows a real Denial of Service attack against a web server. It is possible to observe that the attacker (172.16.114.50) floods the server (135.13.216.191) through port 80 trying to overthrow the services provided. According to the IDS, each line (1, 2, 3, 4, 5 and 7) generates an alarm. However, all the alarms constitute a single attack. This exemplifies a relevant problem for alarm management, the large number of alarms generated.

Table 3: Example of alarms in the data set

N	*Timestamp*	*Src IP*	*Src Port*	*Dst IP*	*Dst Port*	*Packet size*	*Number packets*	*Time interval*	*Δ-variation*	*Attack type*	*Alarm*
1	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	124	back	1
2	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	103	back	1
3	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	61	back	1
4	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	59	back	1
5	920974021	172.16.114.50	80	135.13.216.191	29514	84	20	1	35	back	1
6	920974021	135.13.216.191	29514	172.16.114.50	80	32	59	1	34	-	0
7	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	27	back	1

The evaluation is based on a comparison between the alarms generated for an IDS applied to the DARPA dataset. As general results, we observe that for all attacks fewer alarms were generated after using the *Triple-Similarity Mechanism*, as illustrated by Table 4. The Portsweep attack has shown the highest gain in comparison with other attacks (90%). In other words, the Portsweep attack triggered 574 alarms when using just the IDS. The number of alarms was about 10 times lower than expected when we combine the IDS and T-SyM. The DoS and Ipsweep attacks present similar results, 78.47% and 79.35% fewer alarms, respectively. Finally, our mechanism has decreased the number of alarms from 169 to 44 for the brute force attack.

Along with the aggregation of true positive alarms reported above, our scheme is also able to aggregate false positives, therefore reducing their impact on the IDS performance. In particular, the IDS originally reported an amount of 2632 false alarms. After using T-SyM, by aggregating false alarms, their number decreased to 490, corresponding to an improvement of 81.4%.

The IDS is subject to failures. For example, a missing alarm could occur in the interval between two (or more) attacks that are performed at the same

Table 4: Comparison between alarms generated with and without the Similarity Mechanism

Type of attack	Number of alarms		Improvement (%)
	IDS	IDS + T-SyM	
Brute Force	169	44	73.96%
Portsweep	574	57	90.07%
Ipsweep	155	32	79.35%
DoS	641	138	78.47%

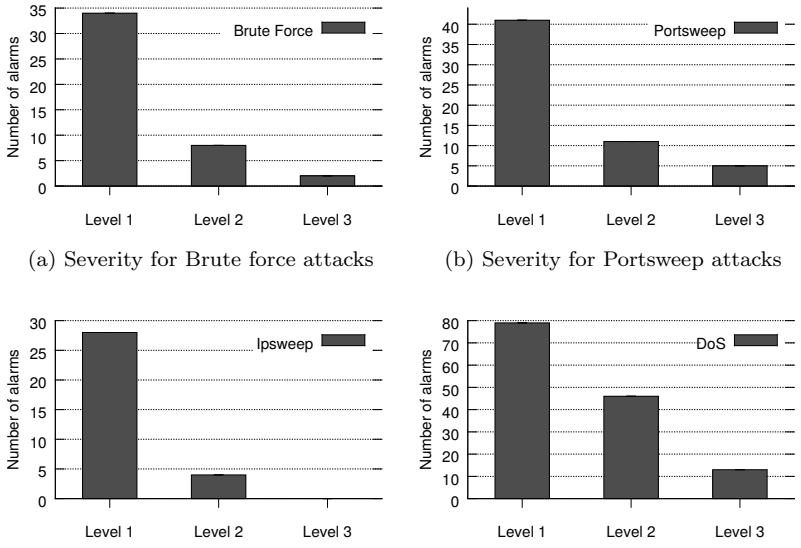
time. However, the problem of missing alarms is the exclusive responsibility of the IDS in question. Once the IDS generates alarms, they can be aggregated by the Triple-Similarity Mechanism. Therefore, the process of aggregation will never generate a missing alarm.

This case study applies Algorithm 1 to calculate the severity level for alarms. This algorithm resorts to a function ω for assessing the severity level of alarms based on a regression model fed with the history of attacks in the *Alarms Database*. We approximate the ω function to a logarithm base 4, meaning that an alarm at level 3 (the greatest risk for the system in this example) should aggregate, at least, 64 alarms from the IDS. Each level corresponds to a different class of alarm priority in increasing order of risk. In this case, actions can be generated and automatically executed in response to a specific alarm level. Figure 3 illustrates the severity level for the alarms. In general, most of the alarms are regarded as level 1 (about 66% of the alarms), 26% are level 2, whilst level 3 alarms represent 8% of the total.

Moreover, the resulting dataset regarding the second week for the *training phase* has more than 400MB of data. After using our mechanism, the total amount of data was 78.3MB. In particular, our mechanism was able to decrease the number of alarms generated by aggregating similar alarms from the IDS. In addition, it is efficient when recognizing the attacks from less information.

It is worth noting that the T-SyM is not evaluating the effectiveness of the IDS to detect attacks nor the diversity of the labelled attacks contained within the DARPA dataset. Although the DARPA dataset was not designed for the cloud, it has a wide range of attacks that make it possible to detect threats from multiple hosts. In this context, using this dataset does not limit the proposal presented here. In fact, the DARPA dataset provides all the conditions necessary to validate our mechanism regarding distributed attacks, commonly present in the cloud. Therefore, this scenario expresses a case study for a mechanism designed to aggregate alarms in a cloud environment.

Lastly, the T-SyM has proved to be efficient at aggregating alarms that carry similar features. In comparison with the alarms generated only by the IDS, our similarity approach produces fewer alarms but with higher levels of severity. This makes the network traffic monitoring of the cloud providers faster and more effective.



(a) Severity for Brute force attacks

(b) Severity for Portsweep attacks

(c) Severity for Ipsweep attacks

(d) Severity for DoS attacks

Figure 3: Evaluation of the severity for alarms

6. Final Considerations and Future Work

In this paper, we present the main issues generated by Intrusion Detection Systems for cloud computing. For instance, the huge number of alarms generated over time and how this impacts on the number of control messages between virtual machines and servers. To address these problems, we propose a Triple-Similarity Mechanism, a systematic approach to aggregate similar alarms in the context of the cloud network traffic and an algorithm to assign severity level for alarms.

From the observation of the results, we can see that our mechanism was able to (*i*) reduce the generation of alarms by from 73% to 90% and; (*ii*) decrease the network data traffic to manage IDS and its associated transfer costs by more than 80%. Moreover, aggregating similar alarms produces fewer alarms but with higher levels of severity, supporting the network traffic monitoring of the cloud providers. Prospects for future research include extending the evaluation in another two directions: (i) testing other IDSs and datasets to ensure the generalised application of our mechanism; and (ii) adapting the T-SyM to operate online.

Acknowledgment

This work was partially funded by CAPES and CNPq (Brazil) through the Ciência sem Fronteiras Program/2017.

References

[1] J. W. Rittinghouse, J. F. Ransome, Cloud computing: implementation, management, and security, CRC press, 2016.

[2] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28 (3) (2012) 583 – 592. doi:http://dx.doi.org/10.1016/j.future.2010.12.006.
URL http://www.sciencedirect.com/science/article/pii/S0167739X10002554

[3] P. Mell, T. Grance, The nist definition of cloud computing, NIST special publication 800 (2011) 1–7.

[4] M. Ali, S. U. Khan, A. V. Vasilakos, Security in cloud computing: Opportunities and challenges, Information Sciences 305 (2015) 357 – 383. doi:http://dx.doi.org/10.1016/j.ins.2015.01.025.
URL http://www.sciencedirect.com/science/article/pii/S0020025515000638

[5] A. Hudic, P. Smith, E. R. Weippl, Security assurance assessment methodology for hybrid clouds, Computers & Security 70 (Supplement C) (2017) 723 – 743. doi:https://doi.org/10.1016/j.cose.2017.03.009.
URL http://www.sciencedirect.com/science/article/pii/S0167404817300627

[6] H. Ballani, P. Costa, T. Karagiannis, A. I. Rowstron, Towards predictable datacenter networks., in: SIGCOMM, Vol. 11, 2011, pp. 242–253.

[7] D. Plonka, P. Barford, Network anomaly confirmation, diagnosis and remediation, in: 47th Annual Allerton Conference on Communication, Control, and Computing, 2009. Allerton 2009., IEEE, 2009, pp. 128–135. doi:10.1109/ALLERTON.2009.5394858.

[8] S. Benferhat, A. Boudjelida, K. Tabia, H. Drias, An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge, Applied Intelligence 38 (4) (2013) 520–540. doi:10.1007/s10489-012-0383-7.
URL http://dx.doi.org/10.1007/s10489-012-0383-7

[9] Z. Li, G. Xia, H. Gao, Y. Tang, Y. Chen, B. Liu, J. Jiang, Y. Lv, Netshield: massive semantics-based vulnerability signature matching for high-speed networks, ACM SIGCOMM Computer Communication Review 40 (4) (2010) 279–290. doi:10.1145/1851275.1851216.
URL http://doi.acm.org/10.1145/1851275.1851216

[10] C.-C. Lo, C.-C. Huang, J. Ku, A cooperative intrusion detection system framework for cloud computing networks, in: 39th International Conference on Parallel Processing Workshops (ICPPW), 2010, 2010, pp. 280–284. doi:10.1109/ICPPW.2010.46.

[11] A. Patel, M. Taghavi, K. Bakhtiyari, J. C. Junior, An intrusion detection and prevention system in cloud computing: A systematic review, Journal of Network and Computer Applications 36 (1) (2013) 25–41. doi:http://dx.doi.org/10.1016/j.jnca.2012.08.007.
URL http://www.sciencedirect.com/science/article/pii/S108480451200183X

[12] H. T. Elshoush, I. M. Osman, Alert correlation in collaborative intelligent intrusion detection systems - a survey, Applied Soft Computing 11 (7) (2011) 4349 – 4365, soft Computing for Information System Security. doi:http://dx.doi.org/10.1016/j.asoc.2010.12.004.
URL http://www.sciencedirect.com/science/article/pii/S156849461000311X

[13] N. Hubballi, V. Suryanarayanan, False alarm minimization techniques in signature-based intrusion detection systems: A survey, Computer Communications 49 (2014) 1–17. doi:http://dx.doi.org/10.1016/j.comcom.2014.04.012.
URL http://www.sciencedirect.com/science/article/pii/S0140366414001480

[14] R. Di Pietro, L. V. Mancini, Intrusion detection systems, Vol. 38, Springer Science & Business Media, 2008.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A berkeley view of cloud computing, Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009).
URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[16] J. Arshad, P. Townend, J. Xu, A novel intrusion severity analysis approach for clouds, Future Generation Computer Systems 29 (1) (2013) 416–428, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. doi:http://dx.doi.org/10.1016/j.future.2011.08.009.
URL http://www.sciencedirect.com/science/article/pii/S0167739X11001488

[17] K. Vieira, A. Schulter, C. Westphall, C. Westphall, Intrusion detection for grid and cloud computing, IT Professional 12 (4) (2010) 38–43. doi:10.1109/MITP.2009.89.

[18] S. Fu, Performance metric selection for autonomic anomaly detection on cloud computing systems, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, 2011, pp. 1–5. doi:10.1109/GLOCOM.2011.6134532.

[19] D. Parikh, T. Chen, Data fusion and cost minimization for intrusion detection, IEEE Transactions on Information Forensics and Security 3 (3) (2008) 381–389.

[20] L. Y. Beng, S. Ramadass, S. Manickam, T. S. Fun, A survey of intrusion alert correlation and its design considerations, IETE Technical Review 31 (3) (2014) 233–240. doi:10.1080/02564602.2014.906864.
URL http://dx.doi.org/10.1080/02564602.2014.906864

[21] H. T. Elshoush, I. M. Osman, An improved framework for intrusion alert correlation, in: Proceedings of the World Congress on Engineering, Vol. 1, 2012, pp. 1–6.

[22] M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita, Network anomaly detection: methods, systems and tools, Ieee communications surveys & tutorials 16 (1) (2014) 303–336.

[23] A. Lakhina, M. Crovella, C. Diot, Characterization of network-wide anomalies in traffic flows, in: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM, 2004, pp. 201–206.

[24] A. Kind, M. Stoecklin, X. Dimitropoulos, Histogram-based traffic anomaly detection, IEEE Transactions on Network and Service Management 6 (2) (2009) 110–121. doi:10.1109/TNSM.2009.090604.

[25] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set, Information Security Journal: A Global Perspective 25 (1-3) (2016) 18–31.

[26] B. L. Dalmazo, J. P. Vilela, P. Simoes, M. Curado, Expedite feature extraction for enhanced cloud anomaly detection, in: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 1215–1220. doi:10.1109/NOMS.2016.7502990.

[27] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, 2005, pp. 539–546 vol. 1. doi:10.1109/CVPR.2005.202.

[28] M. Mohler, R. Mihalcea, Text-to-text semantic similarity for automatic short answer grading, in: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2009, pp. 567–575.

[29] R. L. Cilibrasi, P. M. B. Vitanyi, The google similarity distance, IEEE Transactions on Knowledge and Data Engineering 19 (3) (2007) 370–383. doi:10.1109/TKDE.2007.48.

[30] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, City 1 (2) (2007) 1.

[31] L. Leydesdorff, On the normalization and visualization of author co-citation data: Salton's cosine versus the jaccard index, Journal of the American Society for Information Science and Technology 59 (1) (2008) 77–85. doi:10.1002/asi.20732.
URL http://dx.doi.org/10.1002/asi.20732

[32] B. L. Dalmazo, J. P. Vilela, M. Curado, Performance analysis of network traffic predictors in the cloud, Journal of Network and Systems Management 25 (2) (2017) 290–320. doi:10.1007/s10922-016-9392-x.
URL http://dx.doi.org/10.1007/s10922-016-9392-x

[33] Q. Yan, F. R. Yu, Q. Gong, J. Li, Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Communications Surveys Tutorials 18 (1) (2016) 602–622. doi:10.1109/COMST.2015.2487361.

[34] S. Chatterjee, A. S. Hadi, Regression analysis by example, John Wiley & Sons, 2015.

[35] J. Haines, L. Rossey, R. Lippmann, R. Cunningham, Extending the darpa off-line intrusion detection evaluations, in: DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings, Vol. 1, 2001, pp. 35–45 vol.1. doi:10.1109/DISCEX.2001.932190.

[36] W. Xiong, H. Hu, N. Xiong, L. T. Yang, W.-C. Peng, X. Wang, Y. Qu, Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications, Information Sciences 258 (2014) 403–415. doi:http://dx.doi.org/10.1016/j.ins.2013.04.009.
URL http://www.sciencedirect.com/science/article/pii/S0020025513002971

[37] P. Ganeshkumar, N. Pandeeswari, Adaptive neuro-fuzzy-based anomaly detection system in cloud, International Journal of Fuzzy Systems 18 (3) (2016) 367–378. doi:10.1007/s40815-015-0080-x.

[38] Y. Liu, K.-K. Tseng, J.-S. Pan, Statistical based waveform classification for cloud intrusion detection, in: 2012 International Conference on Computing, Measurement, Control and Sensor Network (CMCSN), 2012, pp. 225–228. doi:10.1109/CMCSN.2012.118.

[39] A. Y. Nur, M. E. Tozal, Record route ip traceback: Combating dos attacks and the variants, Computers & Security 72 (Supplement C) (2018) 13 – 25. doi:https://doi.org/10.1016/j.cose.2017.08.012.
URL http://www.sciencedirect.com/science/article/pii/S0167404817301773