Spotting Intrusion Scenarios from Firewall Logs Through a Case-Based Reasoning Approach

Fábio Elias Locatelli, Luciano Paschoal Gaspary, Cristina Melchiors, Samir Lohmann, and Fabiane Dillenburg

Programa Interdisciplicar de Pós-Graduação em Computação Aplicada (PIPCA)
Universidade do Vale do Rio dos Sinos (UNISINOS)
Av. Unisinos 950 – 93.022-000 – São Leopoldo – Brazil
paschoal@exatas.unisinos.br

Abstract. Despite neglected by most security managers due to the low availability of tools, the content analysis of firewall logs is fundamental (a) to measure and identify accesses to external and private networks, (b) to access the historical growth of accesses volume and applications used, (c) to debug problems on the configuration of filtering rules and (d) to recognize suspicious event sequences that indicate strategies used by intruders in attempt to obtain non-authorized access to stations and services. This paper presents an approach to classify, characterize and analyze events generated by firewalls. The proposed approach explores the case-based reasoning technique, from the Artificial Intelligence field, to identify possible intrusion scenarios. The paper also describes the validation of our approach carried out based on real logs generated along one week by the university firewall.

1 Introduction

The strategy of using a firewall as a border security mechanism allows the centralization, in only one machine, of all the traffic coming from the Internet to the private network and vice-versa. In this control point, any packet (HTTP, FTP, SMTP, SSH, IMAP, POP3, and others) that comes in and out is inspected and can be accepted or rejected, according to the established security rules.

In this context, firewalls store – for each successful or frustrated attempt – records in log files. Some recorded data are: type of operation, source and destination network addresses, local and remote ports, among others. Depending on the network size and its traffic, the daily log can be greater than 1GB [7]. From the security management point of view, this log is rich in information because it allows: (a) to measure and identify the accesses to the private and external networks (e.g. most and least required services, stations that use more or less bandwidth, main users); (b) to historically follow the growth of the accesses and the applications used; (c) to debug problems on filtering configuration rules; and (d) to recognize suspicious event sequences that indicate strategies used by intruders trying to obtain improper access to stations and services.

At the same time that the importance of these indicators is recognized, the growth of information transiting every day between the private network and the Internet has turned the manual control of the log files unviable. This paper presents an approach to classify, characterize and analyze firewall events. The paper describes, yet, the validation of the approach based on real logs generated during one week by the university firewall. The contributions of this work can be unfolded in two: (i) the approach allows identification of sequences of actions executed from or to a determined service or station through the grouping of related events; (ii) supported by the Artificial Intelligence technique called case-based reasoning, the approach provides conditions so that intrusion scenarios[1] can be modeled as cases; whenever similar sequences are repeated, the approach is able to identify them and notify the manager.

The paper is organized as follows: section 2 describes related work. Section 3 presents the proposed approach to classify and characterize the firewall events, as well as to identify automatically the intrusion scenarios. Section 4 describes the tool developed and section 5, the case study carried out to validate it. Finally, section 6 ends up the paper with the final considerations and future work perspectives.

2 Related Work

A quantitative characterization of the intrusion activities performed in the global Internet, based on firewall log analysis, was carried out by Yegneswaran in [9]. The work involved the collection, during a four month period, of more than 1.600 firewall and intrusion detection system logs distributed all over the world. The results enabled to characterize different kinds of probes and their relation to viruses and worms dissemination. It is worthwhile mentioning the fact that this work was carried out in an ad hoc way, without any tool support (this compromises a periodic, long-term analysis). Besides, the approach is exclusively quantitative, what turns difficult the comprehension of some situations in which the events need to be analyzed closely to confirm a suspicious activity.

Regarding event analysis, Artificial Intelligence techniques have been applied to relate events generated by security systems [1,4,5]. Ning presents in [4] a method that correlates prerequisites and consequences of alerts generated by intrusion detection systems in order to determine the various attack stages. The authors claim that an attack usually has different steps and it does not happen in isolation, that is, each attack stage is prerequisite to the next. The method is hard to deploy in large scale. First, prerequisites and consequences must be modeled as predicates, which is not an easy task. Second, the cases database needs to be constantly updated, which requires substantial work. Furthermore, the proposal is limited for not being effective to identify attacks where the relation of cause and consequence cannot be established. For example, two attacks (Smurf and SYN flooding) launched almost at the same time against the same target from two different locations would not be related (however there exists a strong connection between them: same instant and same target).

[1] In this paper an *intrusion scenario* is defined as a sequence of suspicious activities that is executed by an intruder in order to obtain non-authorized access to stations and services.

The approaches described in [1,5] analyze alerts produced by spatially distributed heterogeneous information security devices. They propose algorithms for aggregation and correlation of intrusion-detection alerts. The first defines a unified data model for intrusion-detection alerts and a set of rules to process the alerts. The detection algorithm can detect (*i*) alerts that are reported by different probes but are related to the same attack (*duplicates*) and (*ii*) alerts that are related and should occur together (*consequences*). The second approach uses strategies as topology analysis, alert priorization, and common attribute-based alert aggregation. An incident rank calculation is performed using an adaptation of Bayes framework for belief propagation in trees. These approaches tend not to cope well with the detection of intrusion scenarios that differ (even slightly) from what has been previously defined as *fusion* and *aggregation rules*.

Other Artificial Intelligence techniques have been applied to event processing, especially in the context of intrusion detection systems. One of them is the case-based reasoning paradigm (CBR). Schwartz presents in [6] a tool that applies this paradigm to a variation of the intrusion detection system Snort, where each system signature is mapped to a case. Other system that uses the CBR paradigm is presented by Esmaili in [2]. It uses CBR to detect intrusions using the audit logs produced by the operating system. The cases represent intrusion scenarios formed by operating system command sequences that result in an unauthorized access.

3 Approach to Classify, Characterize, and Analyze Firewall Events

This section describes the approach proposed to classify, characterize and analyze firewall events. It is structured in two independent and complementary parts. The first, more quantitative, allows events stored by the firewall to be grouped based on one or more aggregation elements (filters) defined by the security manager. The second part proposes to analyze these events and identify, automatically, intrusion scenarios (supported by the case-based reasoning technique).

3.1 Event Classification and Characterization

As already mentioned in the Introduction, each event generated by a firewall stores important information such as event type, source and destination addresses, local and remote ports, and others. Since some of these information are repeated in more than one type of event, it is possible to group events using one or more aggregation elements. This constitutes the central idea of the first part of the approach.

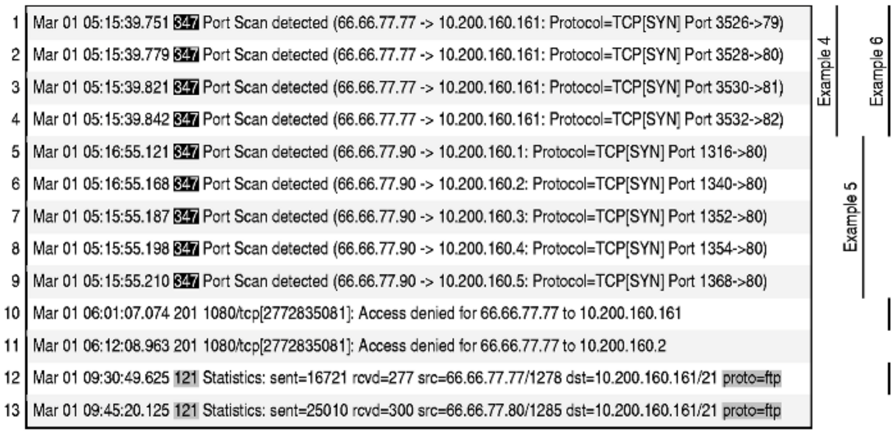
By grouping events that share common information, it becomes possible to perform a series of operations to (a) measure and identify accesses to external and private networks, including malicious actions (port scanning and attempts to access unauthorized services), (b) follow their evolution along the time, (c) debug filtering rules configuration problems, among others. Figure 1 offers many examples in this direction; some of them are commented below.

Example 1. To determine the total data sent and received to FTP connections it is necessary to group the events that belong to the statistical group (121) and that have the field protocol with the value ftp (proto=ftp). This grouping results in the events 12 and 13 (see figure 1). The accounting of the amount of exchanged data is given by the sum of the values associated to the fields sent and rcvd.

Example 2. Inconsistencies and errors in the configuration of filtering rules can be detected with similar grouping. Consider that the organization's security policy establishes that the FTP service, running in the station 10.200.160.161, must not be accessed by external hosts (IPs out of the range 10.200.160.X). The grouping presented in example 1 highlights two events, 12 and 13, which confirms the violation of such policy, since both accesses come from stations with network prefix 66.66.77.X.

Example 3. The identification of the hosts from where departed the major number of port scans is obtained by grouping 347 events, which results in the sub-group {1,2,3,4,5,6,7,8,9}. Four out of these events indicate probes departing from the station 66.66.77.77 and five from the station 66.66.77.90.

Following the same reasoning, other aggregation elements (or a combination of them) can be employed with the purpose of identifying, among the connections performed through the firewall, maximums and minimums in respect to protocols used, hosts and accessed ports, as well as quantity of hits referring to events such as port scanning and access denied, and stations that suffer and launch more port scans.



```
 1  Mar 01 05:15:39.751 347 Port Scan detected (66.66.77.77 -> 10.200.160.161: Protocol=TCP[SYN] Port 3526->79)
 2  Mar 01 05:15:39.779 347 Port Scan detected (66.66.77.77 -> 10.200.160.161: Protocol=TCP[SYN] Port 3528->80)
 3  Mar 01 05:15:39.821 347 Port Scan detected (66.66.77.77 -> 10.200.160.161: Protocol=TCP[SYN] Port 3530->81)
 4  Mar 01 05:15:39.842 347 Port Scan detected (66.66.77.77 -> 10.200.160.161: Protocol=TCP[SYN] Port 3532->82)
 5  Mar 01 05:16:55.121 347 Port Scan detected (66.66.77.90 -> 10.200.160.1: Protocol=TCP[SYN] Port 1316->80)
 6  Mar 01 05:16:55.168 347 Port Scan detected (66.66.77.90 -> 10.200.160.2: Protocol=TCP[SYN] Port 1340->80)
 7  Mar 01 05:15:55.187 347 Port Scan detected (66.66.77.90 -> 10.200.160.3: Protocol=TCP[SYN] Port 1352->80)
 8  Mar 01 05:15:55.198 347 Port Scan detected (66.66.77.90 -> 10.200.160.4: Protocol=TCP[SYN] Port 1354->80)
 9  Mar 01 05:15:55.210 347 Port Scan detected (66.66.77.90 -> 10.200.160.5: Protocol=TCP[SYN] Port 1368->80)
10  Mar 01 06:01:07.074 201 1080/tcp[2772835081]: Access denied for 66.66.77.77 to 10.200.160.161
11  Mar 01 06:12:08.963 201 1080/tcp[2772835081]: Access denied for 66.66.77.77 to 10.200.160.2
12  Mar 01 09:30:49.625 121 Statistics: sent=16721 rcvd=277 src=66.66.77.77/1278 dst=10.200.160.161/21 proto=ftp
13  Mar 01 09:45:20.125 121 Statistics: sent=25010 rcvd=300 src=66.66.77.80/1285 dst=10.200.160.161/21 proto=ftp
```

Fig. 1. Real event set extracted from a log and their relations

3.2 Automatic Event Analysis

In addition to a more quantitative analysis, where diverse accountings are possible, our approach allows the automatic identification of intrusion scenarios based on the observation of more elementary event groups. In figure 1 three suspicious behaviors can be highlighted and are detailed bellow.

Example 4. The first consists of a vertical port scanning and it is composed of events 1, 2, 3, and 4. This probe is characterized by scans coming from a single IP address to multiple ports of another IP address. Observe that four port scans were launched, in less than one second, from the host 66.66.77.77 to the host 10.200.160.161.

Example 5. The second suspicious behavior comprehends a horizontal port scanning and includes the events 5, 6, 7, 8, and 9. In this case, the probes depart from an IP address to a single port of multiple IP addresses. As it can be observed in figure 1, the probable invader 66.66.77.90 scanned port 80 of several different hosts searching for one that had an HTTP server available.

Example 6. Finally, the third intrusion scenario corresponds to a probe followed by a successful access, including the events 1, 2, 3, 4, 10, and 12. The station 10.200.160.161 suffered four scans (ports 79, 80, 81, and 82) and one unsuccessful access attempt to the port 1080. Both the port scans and the access attempt departed from the host 66.66.77.77 that, at last, obtained access to the station using the FTP protocol (event 12); the elevated number of data sent indicates an upload to the target station (10.200.160.161).

Due to the high number of firewall events, scenarios as the ones mentioned escape, many times, unnoticed by the security manager. The second part of the approach, detailed in this subsection, proposes the use of the case-based reasoning paradigm to identify intrusion scenarios in an automatic way.

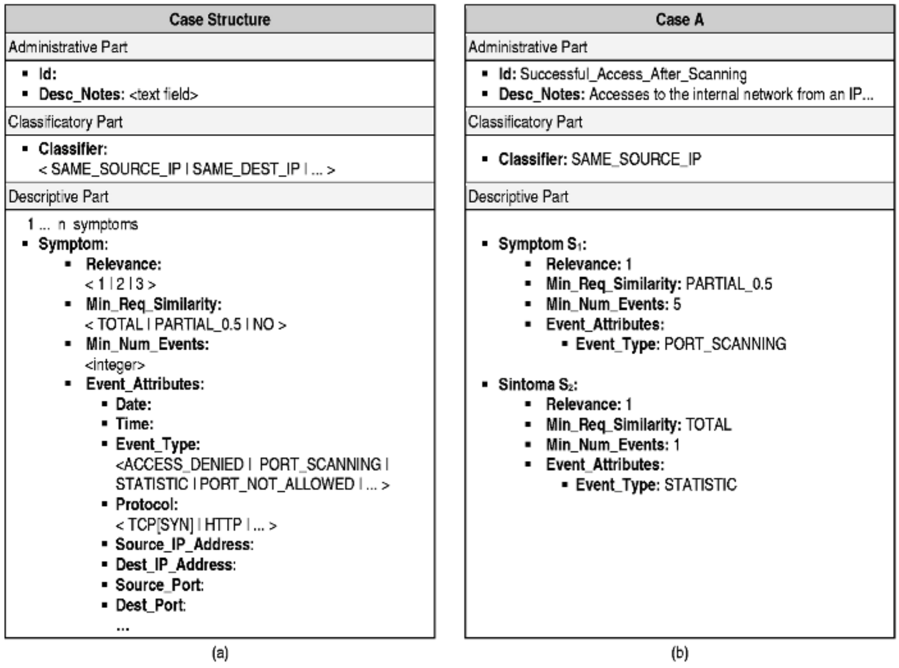
Case-based reasoning (CBR) [3] is an Artificial Intelligence paradigm that uses the knowledge of previous experiences to propose a solution in new situations. The past experiences are stored in a CBR system as cases. During the reasoning process for the resolution of a new situation, it is compared to the cases stored in the knowledge base and the most similar cases are used to propose solutions to the current problem.

The CBR paradigm has some advantages to other reasoning paradigms. One of them concerns to the facility of knowledge acquisition, which is carried out searching real experiences from past situations [2]. Other advantage is the possibility of obtaining partial match between the new situation and the cases, allowing more flexibility in domains where symptoms and problem conditions can have small variation when occurring in real situations.

Case Structure. In our approach a case stored represents a possible intrusion scenario or a suspicious activity that can be identified from the firewall events stored in the log. The case structure is presented in figure 2a. As one can observe, a case is formed by: (a) *administrative part*, with fields for identification and notes that are not used during the reasoning process; (b) *classificatory part*, which contains a field used to divide the log in parts (explained later on); and (c) *descriptive part*, which contains the attributes used to match the cases.

The similarity between the events of the real log and the cases stored is calculated by the presence of events with certain characteristics in the log; we call it a *symptom*. In other words, a symptom is the representation of one or various suspicious events that should be identified in the log so that the stored case can be considered similar to the current situation.

A case can contain one or more symptoms, according to the characteristics of the intrusion scenario or the suspicious activity being described. An example of case with two symptoms is presented in figure 2b. The case modeled, simplified to facilitate the description of the approach, suggests that an alarm should be generated whenever around five scans and a successful access are observed departing from the same source station. Symptom S_1 represents PORT_SCANNING events, such as events 1 to 4 in figure 1, while symptom S_2 represents STATISTIC events, such as the event 12 in the same figure.



Case Structure

Administrative Part

- **Id:**
- **Desc_Notes:** <text field>

Classificatory Part

- **Classifier:**
  < SAME_SOURCE_IP | SAME_DEST_IP | ... >

Descriptive Part

1 ... n symptoms
- **Symptom:**
  - **Relevance:**
    < 1 | 2 | 3 >
  - **Min_Req_Similarity:**
    < TOTAL | PARTIAL_0.5 | NO >
  - **Min_Num_Events:**
    <integer>
  - **Event_Attributes:**
    - **Date:**
    - **Time:**
    - **Event_Type:**
      <ACCESS_DENIED | PORT_SCANNING | STATISTIC | PORT_NOT_ALLOWED | ... >
    - **Protocol:**
      < TCP[SYN] | HTTP | ... >
    - **Source_IP_Address:**
    - **Dest_IP_Address:**
    - **Source_Port:**
    - **Dest_Port:**
    ...

(a)

Case A

Administrative Part

- **Id:** Successful_Access_After_Scanning
- **Desc_Notes:** Accesses to the internal network from an IP...

Classificatory Part

- **Classifier:** SAME_SOURCE_IP

Descriptive Part

- **Symptom S_1:**
  - **Relevance:** 1
  - **Min_Req_Similarity:** PARTIAL_0.5
  - **Min_Num_Events:** 5
  - **Event_Attributes:**
    - **Event_Type:** PORT_SCANNING
- **Sintoma S_2:**
  - **Relevance:** 1
  - **Min_Req_Similarity:** TOTAL
  - **Min_Num_Events:** 1
  - **Event_Attributes:**
    - **Event_Type:** STATISTIC

(b)

Fig. 2. Intrusion scenarios and suspicious activities modeled as cases

Parameters of the log events such as date, time, type of event, and source IP are represented in a case as attributes of the event that composes the symptom. Not all the attributes need to be defined (fulfilled); only the defined ones will be used to calculate the similarities (presented later on). Considering case A illustrated in figure 2b, only the attribute *Event_Type* is being used to identify the event that constitutes the symptom S_1. The same happens to the definition of the symptom S_2.

Reasoning Processes. The matching of log events with a stored case starts by the separation of these events in parts. The criterion to be adopted in this separation is determined by the field *Classifier* (see figure 2a). Each part is called *current case* and is compared to the stored case in a separate way. Take as example the comparison of the log events presented in figure 1 with the case A, figure 2b. Case A has as classificatory attribute the use of a same source IP address (field *Classifier* equal to SAME_SOURCE_IP). Thus, during the reasoning process the example log events are divided in two different cases, one containing the events 1 to 4 and 10 to 12 (which we will call current case 1) and the other containing events 5 to 9 (which we will call current case 2 henceforth).

After the separation of the log events in current cases, as explained above, each current case must be compared to the stored case in order to calculate its similarity, through a process called match between current case and stored case. This match is done using the similarity of the current case events regarding each symptom present in the stored case in a step called symptom matching. Back to case A and to the current case 1 of the previous example, the similarity between them is calculated using the similarity of case A symptoms, which are S_1 and S_2. At last, the similarity of a symptom is calculated based on the similarity of the current case events to the event attributes of that symptom (*Event Attributes*). In the example, the similarity of S_1 is calculated using the similarity of each event of the current case 1 (events 1 to 4 and 10 to 12) to the event attributes of that symptom (field *Event Type* equal to PORT SCANNING). These steps are explained below.

The similarity of a current case event to the event attributes of a symptom of the stored case is calculated by the total sum of each attribute similarity defined in the symptom, divided by the number of defined attributes. The approach allows the similarity of event attributes to be partial or total. In the current version, only similarities of event attributes that assume total (1) or no (0) match have been initially modeled. Resuming the example of the current case 1 and case A, in the event similarity calculation regarding symptom S_1, there is only one defined attribute, which is the *Event_Type*. The similarity of the events 1 to 4 results in 1 (100%), since these events are of the PORT_SCANNING type, which is the same event type defined in the attribute *Event_Type*. On the other hand, the similarity of the events 10 to 12 results in 0, because these events are not of PORT_SCANNING type. Considering now the similarity of the symptom S_2, there is also only one attribute defined (type of event). In the calculation of similarity of each event of the current case 1 in respect to the symptom S_2, the events 1 to 4, 10 and 11 result in 0, while the similarity of event 12 results in 1 (field *Event_Type* equal to STATISTIC).

After the calculation of the events similarity in respect to a symptom, they are ordered by their similarity. The *n* events with higher similarity are then used to match the symptom, where *n* indicates the minimum number of events needed to have total similarity to that symptom (modeled in the case as *Min_Num_Events*). The similarity of the symptom is calculated by the sum of the similarity of these *n* events divided by *n*. If the resulting similarity for a symptom is under the minimum similarity defined for that symptom in the stored case (modeled by *Min_Req_Similarity*), the comparison of that current case with the stored case is interrupted, and the current case is discarded. Recalling the previous example, the event ordering for symptom S_1 results in {1, 2, 3, 4, 10, 12}. As for this symptom the minimum number of events to total match is 5, its similarity will be calculated by $(1 + 1 + 1 + 1 + 0)/5 = 0.8$. Since the minimum similarity defined in the case for symptom S_1 is 0.5, this symptom is accepted and the process continues, calculating the similarity of the other symptoms in the case (S_2 in the example). Considering now symptom S_2 that has *Min_Num_Events* equals to 1, the similarity is calculated by $(1)/1 = 1$. With similarity 1, S_2 is also accepted.

Finally, after matching all the symptoms in the stored case, the match of the current case and the stored case is performed. This calculation is done considering the symptom similarity and its relevance using the formula bellow; *ns* is the number of symptoms of the stored case, r_i is the relevance of symptom *i* and $symptom\_sim_i$ is the similarity of symptom *i*. Referring once more the current case 1 and case A, the final match degree will be ((1 × 0.8) + (1 × 1))/2 = 0.9, 90%. In this example, both symptoms have the same importance (*Relevance*), but assigning different weights can be necessary in other situations.

$$\frac{\sum_{i=1}^{ns} r_i \times symptom\_sim_i}{\sum_{1}^{ns} r_i}$$

When the similarity degree between the current case and a stored case is higher than a predefined value, the current case is selected as suspicious, indicating a situation that should be reported to the security manager. When a case is selected, some additional parameters are instantiated with data of the current case, in an adaptation process, in order to be possible to provide the manager with a detailed view of the identified problem. An example is the instantiation of the attribute source IP address for the cases in which the classifier corresponds to SAME_SOURCE_IP, as in case A. Using this instantiation, in the example of current case 1 commented during this section, the suspicious attitude could be presented as *Successful_Access_After_Scanning* detected to the source IP address 66.66.77.77.

In addition to the example described above we have modeled several other intrusion scenarios, including *horizontal*, *vertical*, *coordinated*, and *stealth* scans [9], IP spoofing, suspect data uploads, web server attacks, and long-term suspect TCP connections, to mention just a few. These scenarios enabled us to explore more functionalities of the case structure such as alternatives, non-ordered lists of symptoms, and time correlation between symptoms.

4 The SEFLA Tool

To validate the approach we have developed SEFLA (Symantec Enterprise Firewall Log Analysis) tool. It was developed under GNU/Linux environment, using Perl and PHP programming languages, the Apache web server and the MySQL database. Figure 3 illustrates the SEFLA architecture including its components and the interactions among them. The parser module is responsible for processing the log files (1) and inserting the main attributes of each event (e.g. type of operation, source and destination network addresses, local and remote ports, among others) in the database (2). From any web browser the security manager interacts with the core of the tool that was implemented in a set of PHP scripts (3, 4). This interaction allows (a) defining processing configurations (e.g. history size in days and types of events to be analyzed), (b) retrieving reports, (c) querying and visualizing results, (d) watching alerts for intrusion scenarios or suspicious activities and (e) verifying specific event details. For such, the database is always queried or updated (5).

Each type of event is stored in a distinct table. Some attributes, for being common for two or more events, are repeated in the corresponding tables. This scheme was adopted in detriment of a normalized one because in the latter it would require an average of six queries and seven insertions for each event to be inserted in the database (compromising the performance of the processing phase).

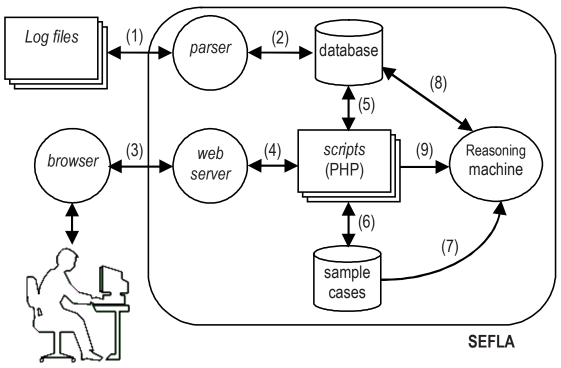


Fig. 3. SEFLA internal components

Through the web browser the security manager also includes, removes and updates cases in the cases database (3, 4, 6), as well as configures functioning parameters for the reasoning machine (3, 4, 9). The identification of intrusion scenarios is done automatically after the tool populates the database with the current day log events (parser module). The reasoning machine then searches in the database for events of interest (8) and confronts them with the sample cases (7). Whenever a new suspicious behavior is identified, the module includes an alarm in the database (8), which will become visible to the security manager.

5 Case Study

The academic network of Universidade do Vale do Rio dos Sinos was used as a case study, whose infrastructure has approximately 4.100 computers connected to it and with Internet access. Log files were collected during a one week period from the firewall located in the border of this network. SEFLA was populated with these logs and through the analysis of the obtained reports it was possible to classify, characterize and analyze the events in order to determine the network use and identify intrusion scenarios and suspicious activities. The tool was installed in an IBM NetVista station, with a 1.8GHz Intel Pentium4 processor, 256MB of RAM and GNU/Linux operating system (Red Hat Linux 9.0 distribution) with a Linux kernel version 2.4.20.

Table 1 describes the profile of each log and its processing characteristics. The largest logs are the ones generated between Monday and Friday. Given the total sum of the size of all log files (13.05GB) and considering that from this volume 52.2% of the events were processed, one can verify that the size of the log file was very reduced when inserted into the database (resulted in 22.4% of the original size). Besides, the time needed to process the 13.05GB of log data was of 144.5 minutes (2 hours, 24 minutes and 30 seconds).

Table 1. Log file sizes and processing time

	LS	PE	PT	ADS
28/09	0,69	1,30	8,1	0,15
29/09	2,53	3,84	28,7	0,72
30/09	2,55	3,79	28,4	1,27
01/10	2,37	3,52	24,0	1,82
02/10	2,28	3,58	23,6	2,31
03/10	1,93	3,10	22,6	2,75
04/10	0,70	1,40	9,1	2,92
Totals	13,05	20,53	144,5	2,92

LS: Log Size (GB), PE: Processed Events (millions),
PT: Processing Time (minutes), ADS: Accumulated Database Size (GB)

Figure 4 illustrates some discoveries, more of quantitative nature, carried out with SEFLA support. In (a) it is presented the data flow through the private and external networks. As it is possible to observe, the HTTP protocol was the most used, followed by TCP/1500 (used by a backup tool), FTP, SMTP, and HTTPS. The total bytes transferred through the networks of more than 30GB from Monday to Friday is another information that deserves to be emphasized. Regarding port scans, data from the day with most occurrences of this event have been processed – in this case, Sunday (see figure 4b). The five stations from where departed the major number of probes have the same network prefix (200.188.175.X). When such a hostile behavior is identified, requests coming from this network addresses should be carefully analyzed (or blocked by the firewall). Figure 4c, in counterpart, highlights stations that were most targeted for port scanning in the analyzed week. Still on the port scan analysis, figure 4d illustrates the history of the most probed port. According to the study performed about the logs, the destination port 135 represented 90% of the total probes in the period of seven days. This port is commonly used under Windows platform to start an RPC (Remote Procedure Call) connection with a remote computer. The port scans observed are probably due to the worms W32.Blaster.Worm and W32.Welchia.Worm released, respectively, in 11/Aug/2003 and 18/Aug/2003. These worms are characterized for exploring an RPC vulnerability in the DCOM (Distributed Component Object Model) acting through the TCP port 135 to launch DoS attacks [8].

Besides the analysis described above, the events collected by the firewall during the week have also been analyzed from the point of view of automatic detection of intrusion scenarios. One of the identified scenarios was the port scan (with similar behavior to the examples 4 and 5), which repeated several times in the log. One instance of this scenario corresponds to the probe represented by 24 port scan events departing from the same source IP 200.226.212.151 to the same destination IP 200.188.160.130 observed on Sunday, 1:48am. This scenario was considered 100% similar to the case *Port_Scan*, as its occurrence involved more than five events of the port scan type originating from the same source IP address (symptom defined for this case). Another scenario recognized in many occasions was the one which comprises port scans and a successful access departing from the same source station as specified in case *Successful_Access_After_Scanning* (figure 2b).

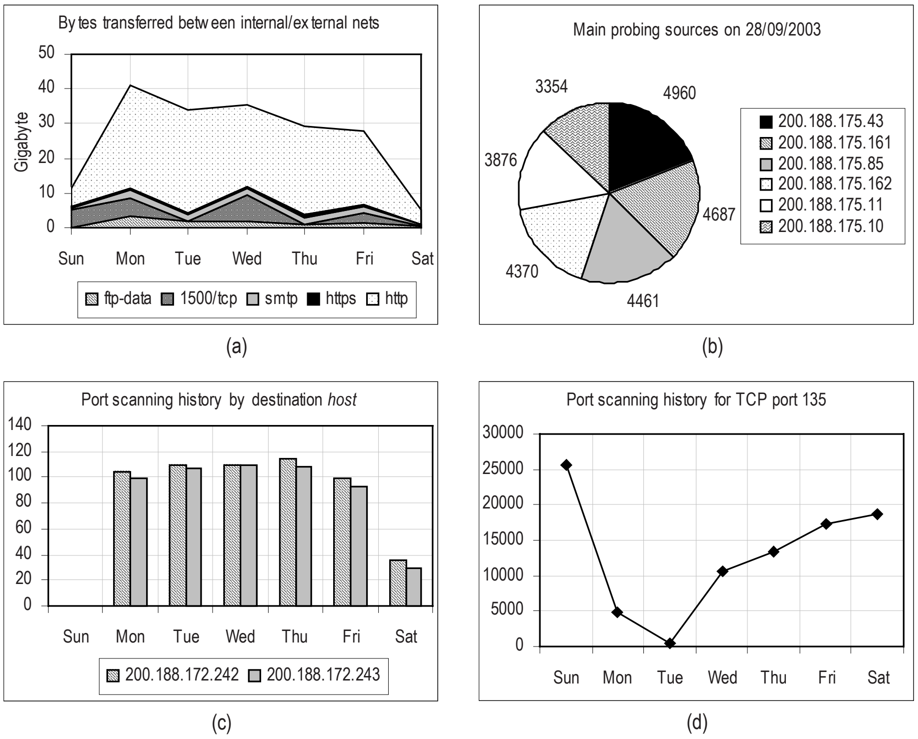


Fig. 4. Some of the information retrieved with the use of SEFLA

6 Conclusions and Future Work

Ensuring the safety of information kept by organizations is a basic requirement for their operation, since the number of security incidents grows exponentially every year. However, to protect organizations considering the quantity and the growing complexity of the executed attacks, it is needed to provide the security manager with techniques and tools that support the analysis of the evidences and, furthermore, allow the automatic identification of intrusion scenarios or suspicious activities. In this context, we presented an approach, accompanied by a tool, for classification, characterization and analysis of events generated by firewalls. It is worth mentioning that our approach does not replace other tools, as the intrusion detection systems, and must be used in conjunction with them.

The organization of the approach in two parts allows handling, in a satisfactory way, both quantitative and qualitative information. On one side, the event grouping mechanism based on one or more aggregation elements reveals network usage characteristics and malicious activities. These can be used (a) to evaluate the accomplishment of the security policy, (b) to control resource usage (reviewing current filtering rules) and (c) to recognize sources and targets of hostile behaviors (aiming at their protection). On the other side, the second part of the approach - supported by the case-based reasoning technique - provides the automatic recognition of event sequences that represent intrusion scenarios or suspicious activities. Here, more than identifying and quantifying actions, one pursuits to recognize the strategies adopted by intruders to obtain unauthorized access to stations, services and applications.

As it could be observed in section 5, even after the processing and storage of the events in the database, the resulting base size is large (considering that it contains events from only seven days). In order to obtain long-term statistics, the synthesis of essential information about the older events is proposed as a future work (at the cost of losing the possibility of detailing these events). Currently, we are working on the evaluation of how much choices on values for *Relevance* and *Min_Req_Similarity*, for example, influence the generation of high-level alerts. From this investigation we expect to learn how to better determine weights for the different parameters in the model.

References

1. Debar, H. and Wespi, A. (2001). Aggregation and correlation of intrusion-detection alerts. Recent Advances in Intrusion Detection, 2212:85-103.
2. Esmaili, M. and et al (1996). Case-Based Reasoning for Intrusion Detection. Computer Security Applications Conference, p. 214-223.
3. Kolodner, J. (1993). Case-Based Reasoning. Morgan Kaufmann.
4. Ning, P., Cui, Y., and Reeves, D. (2002). Analyzing Intensive Intrusion Alerts via Correlation. Recent Advances in Intrusion Detection, 2516:74-94.
5. Porras, P. A. , Fong, M. W. , and Valdes, A. (2002). A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. Recent Advances in Intrusion Detection, 2516:95-114.
6. Schwartz, D., Stoecklin, S., and Yilmaz, E. (2002). A Case-Based Approach to Network Intrusion Detection. Internacional Conference on Information Fusion, p. 1084-1089.
7. Symantec (2001). Symantec Enterprise Firewall, Symantec Enterprise VPN, and VelociRaptor Firewall Appliance Reference Guide. Symantec.
8. Symantec (2003). Symantec Security Response.
9. Yegneswaran, V., Barford, P., and Ulrich, J. (2003). Internet Intrusions: Global Characteristics and Prevalence. ACM SIGMETRICS Performance Evaluation Review, 31(1):138-147.