An Incident Response Support System

Gianluca Capuzzi,† Egidio Cardinale††, Ivan Di Pietro††† and ***Luca Spalazzi****††††,*

Università Politecnica delle Marche, Via Brecce Bianche, 1, Ancona, 60131, Italy

Summary
Computer and network security can be improved by three kinds of tools: tools for intrusion prevention, tools for intrusion detection, and tools for incident response. Many systems have been proposed and developed for the first two kinds of tools. Concerning the third, as far as we know, the response plan is still left to the security manager: no automatic tools have been developed. Indeed, even if there exist forensic analysis, data recovery, and system upgrading tools, we do not yet have a comprehensive tool which includes log correlation, attack classification, and response plan generation. Our work deals with a Case-Based Reasoning system (called IRSS) that classifies attacks, looks in a case base for past attacks similar to the current one (according to given similarity metrics), and reuses the past response plans (adapted to the current attack) in order to restore normal conditions and improve network security. This paper provides an overview of the system and primarly focuses on the incident retrieval (attack classification) phase.

Key words:
IDS, Network Security, Attack Recognition.

Introduction

Computer and network security can be improved by three kinds of tools: tools dealing with prevention (e.g., firewall), tools dealing with detection (e.g., intrusion detection systems - IDS), tools dealing with response. Many systems have been proposed and developed for the first two kinds of tools. Concerning the third one, there are a lot of tools that a security manager can use for incident response: tools for recovering compromised data (e.g., back-up tools), tools for upgrading system security (e.g., patch management tools, intrusion prevention systems), tools for removing the attack (e.g., system management tools, anti-viruses). Nevertheless, as far as we know, no automatic tools for planning a response and for coordinating all the previous tools have been proposed. Which, when, and how the previous tools must be used to respond to an attack is still left to the security manager's past experience. The related work on this topic is still limited to the definition of criteria and policies that must be applied by security managers (e.g., see [5]). Therefore, the aim of our work is focused on tools for incident response planning. Incident response can be defined as *the detection and the identification of an attack to a computer system, the implementation of appropriate responsive actions until normal conditions have been restored*. Therefore, the tool we propose (that we call *Incident Response Support System - IRSS*) must be integrated with other tools dealing with security as firewalls, IDSs, (web, ssh, ...) servers, and so on.

First of all, let us define some concepts we use in the rest of the paper: An **event** is a detectable atomic action performed by an attacker against a given target; e.g., a TCP SYN packet sent to a host. An **attack** is a sequence of events; e.g., a SYN Flooding (sending a great number of packets to a single port). In our approach, the incident response can be planned in three distinct phases. The *first phase* deals with *intrusion detection*. This means collecting data from several sensors on the network and on computers, e.g., log files of operating systems and system servers, firewalls, (network-, host-, application-based) IDSs. The *second phase* deals with *incident assessment* (alarm correlation). This means correlating all the data collected in the previous phase to the end of providing an attack description in terms of sequence of events as complete as possible. The *third phase* deals with *planning* a response to attacks. These plans must contain the following kinds of action:

•Actions for collecting more data about attacks. This implies the possibility of managing sub-goals and corresponding sub-plans (hierarchical planning).

• Actions for restoring normal conditions (e.g., for recovering the compromised data).

• Actions for improving the security level of the system (e.g., feedback to firewalls, NIDS, and HIDS, feedback to security manager, patch management, and so on).

• Actions for communicating with all the involved parties in order to inform them of the attack.

In our approach, the response planning can be accomplished by means of case-based reasoning. In the case memory we have the past attacks and their corresponding response plans. This allows us to have a tool capable of reacting based on previous experience, eventually modifying previous plans, and learning new response plans to even new kinds of attacks. The architecture of IRSS has been proposed in [20].

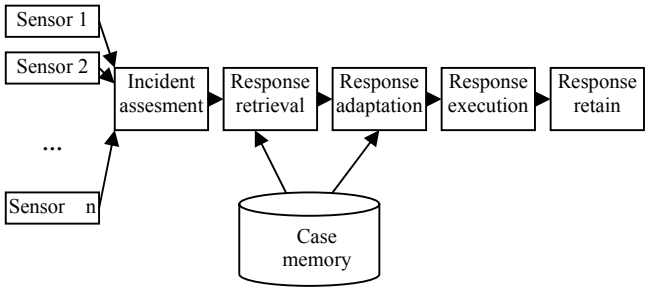

Fig. 1 The architecture of the IRSS.

This paper principally focuses on the incident retrieval phase of the case-based reasoner and on the experimental result about this phase. Indeed, this is the part of the system that search in the case memory for the closest past case to the current sequence of correlated events. This part is obviously useful for finding a response plan that can be used, after an appropriate adaptation, for responding to the current attack, but it is very important for recognizing the kind of attack (attack classification), as well. The paper is structured as follows. Section 2 provides a system overview and the related work. The case memory is described in Section 3. The next sections deal with the specific modules of IRSS: namely, Section 4 deals with Incident Assessment and Response Retrieval, Section 5 with Response Adaptation, Execution and Retainment. In Section 6, we report some experimental results. Some Conclusions are drawn in Section 7.

2. System Overview

Case-based reasoning and planning is known to be extremely useful in exploiting past experience in several application domains (e.g., see [1], especially in the diagnosis and the management of different kinds of emergency [2; 3; 12; 10; 13]. For instance, [2] deals with alarm correlation, but these are fault alarms, not intrusion alarms, and the goal is to obtain a fault tolerant network, not a secure network. As far as we know, only a few examples of CBR applied to network and computer security. All these examples concern the application of CBR to intrusion detection (see [6; 19]). We have just an example of application of CBR to incident response [Nick *et al.*, 2003]. There the goal is to improve detection and to avoid the need of frequently updating the database of known attacks. The internal structure of IRSS is depicted in Figure 1 and follows the standard structure of a case-based reasoner (e.g., see [1]). IRSS includes a case base which contains a set of past incidents (attacks) with their responses (plans). Attacks are represented as event sequences and response plans are represented as partially ordered sets of actions (for our experiments we use Linux scripts). IRSS senses by means of so called *agents* log data from operating systems, web servers, IDSs, and firewalls. These data must be normalized by agents since each log file has its own format. Furthermore, they must be also filtered by agents, since log files of an operating system or a web server contain attack alarms as well as data that do not concern attacks. This is a standard practice for secure systems [4], therefore we do not further describe it. All the data collected by agents are sent to the *incident assessment*

Event Type	Sensor	Source	Target
One to many horizontal scan	NIDS2	207.46.176.50	172.16.113.84:80
WEB-CGI redirect access	NIDS1	207.46.176.50	172.16.113.84:80
WEB-CGI redirect access	NIDS1	207.46.176.50	172.16.113.84:80

Fig. 2 An example of attack.

module. This module is in fact a log correlator. Namely, it is a tool that correlates alarms in order to find a sequence of events representing the same attacks. Alarm correlation is not new and several approaches have been proposed: statistic correlation [11], Bayesian correlation [18], correlation based on pattern matching [17]. Because of this module is based on the work described in [17], we refer the reader to [17] for a detailed description. After that, for each sequence of correleted events is computed the sum of the entropy of all the events in the sequence. Only the sequences with an entropy greater than a given threshold are considered real attacks. This step aims to avoid to respond to very common and not significative sequences as sequences of portscanning or ping. The sequences that are considered real attacks must be compared with the description of past attacks in order to retrieve the most similar past attack of the current one, and reuse the corresponding past response plan (after an appropriate adaptation). This phase is called *Response Retrieval*. We use four similarity metrics. Two of them are simple but effective similarity metrics based on pattern-matching. The other ones are based on the entropy of attacks. The response retrieval module selects the top ranked case with a similarity greater than a given threshold. The *Response Adaptation* tries to adapt the retrieved response (plan) to the current incident. Notice that, the adaptation process can not be completely automatic. Indeed, the security manager must have the possibility to validate the plan and eventually to modify it, when it is needed. The *Response Execution* subsystem is devoted to execute the adapted and validaded plan, monitor its execution, and receive the feedback from the security manager (that evaluates the execution). Finally, the *Response Retainment* module decides whether the executed response must be retained (and thus the case base must be updated) depending on the received feedback. Notice that in this way, IRSS learns new responses to new kinds of attack.

3. Case Memory

A *case* consists of a pair: attack (i.e., a sequence of events), response (i.e., a partially ordered set of actions). The outcome of the Incident Assessment module should be a sequence of concrete events. For example, Figure 2 reports an example of attack to a Web Server. It consists of three events: the detection of the horizontal scanning, and the detection of two web server attempts. For each event, the report of the current attack includes: attack source and target IP addresses, timestamp, and event descriptions (i.e., the event type). Nevertheless, we cannot use attacks as they are, but we must use their abstractions.

ID	Event Type	Sensor	Source	Target	PLAN
Case 1	One to many horizontal scan	NIDS2	int/ext	any:any	Plan1
	One to many horizontal scan	NIDS2	int/ext	any:any	
	Info FTP bad Login	NIDS2	int	ftpserver:ftpport	
	Info FTP bad Login	NIDS2	int	ftpserver:ftpport	
Case 2	WEB-CGI redirect access	NIDS1	ext	any:any	Plan2
	WEB-CGI redirect access	NIDS1	ext	any:any	
	WEB-CGI redirect access	NIDS1	ext	ftpserver:ftpport	
	WEB-CGI redirect access	NIDS1	ext	ftpserver:ftpport	
Case 3	One to many horizontal scan	NIDS1	int/ext	any:any	Plan3
	WEB-MISC/doc/access	NIDS1	int/ext	any:any	
	WEB-MISC/doc/access	NIDS1	int/ext	ftpserver:ftpport	
	WEB-MISC/doc/access	NIDS1	int/ext	ftpserver:ftpport	
Case 4	Many to one	NIDS2	int/ext	any:any	Plan4
	SNMP Request udp	NIDS2	int/ext	any:any	
	SNMP Request udp	NIDS2	int/ext	any:any	
	SNMP Request udp	NIDS2	int/ext	any:any	

Figure 3: A fragment of the Case Memory.

Therefore, the Incident Assessment transforms a concrete attack in an abstract attack before it sends the attack to the Response Retrieval module. Consider an example, let us suppose that in the past we had an event of the kind *Apache exploit* with a given IP number (say 172.16.113.84) as target address. Let us suppose that the current event is an *Apache exploit* on a different IP number (say 192.168.0.3). It seems quite natural to consider these two events similar (they have the same event type) and to reuse the past response, even if these two events are not identical (they have a different target). *Event abstraction* is the tool to find similarities without taking into account irrelevant details. In short, it consists of substituting some values as source and target with their type and some values as start and end time with a partial order relation. For example, if the IP number 172.16.113.84 is the address of a web server, we can substitute the number with the keyword webserver. Formally, let e = < event_type, source, target, start, end > be an event, then Abs(e) = < event_type, source type, target type, ., .> is the abstraction of e, Type(e) = event type is the event type of e, and T (e) = start is used to define the order relation. Let $I = (e_1, \ldots, e_n)$ be an attack, then Abs(I) = < $Abs(e_1), \ldots, Abs(e_n)$ > is the corresponding abstraction of I and Type(I) = < $Type(e_1), \ldots, Type(e_n)$ > the corresponding sequence of event types of I. In fact, attacks in the case memory are abstract attacks (e.g., see Figure 3). To abstract cases, we used a set of pairs, which associate a description of hosts and servers in the network with their IP addresses. In fact, these pairs represent the network configuration and are stored in a configuration file.

4. Response Retrieval

The Incident Assessment output is a list of abstract attacks. This list contains noise, i.e., non-relevant attacks. These attacks must be removed from the list. Therefore, we use a **filter** before the retrieval module. This filter is based on the well-known notion of *Information Entropy*. Indeed, an event that occurs frequently (e.g. an ICMP echo request) usually is not really dangerous and, thus, its detection provides us a little information. Therefore, for each event type t, we compute the probability p(t) (based on the occurring frequency) that such a type of event occurs and, thus, we can compute its entropy as follows:

$$w_t = -\log_2 p(t) \qquad (1)$$

When we detect a sequence composed only by non-relevant events, we are not facing a real dangerous attack. Therefore, for each sequence of events I = < e_1, ..., e_n > we can compute its entropy as follows:

$$H(I) = -\sum_{i=1}^{n} \log_2 p(Type(e_i)) \qquad (2)$$

The filter eliminates all the attacks I such that H(I) < threshold. Now, the Response Retrieval has to retrieve past attacks similar to the attack that passed the filter. Response retrieval can be achieved by means of four different similarity functions. They are defined as follows:

Definition 1 *Let* $I_c = < e_{c,1}, \ldots, e_{c,n} >$ *be the current attack,let* $I_k = <e_{k,1}, \ldots, e_{k,n}>$ *be the attack of the k-th case in the case memory, let Type(I_c) and Type(I_k) be the sequence of event types of Ic and* I_k*, respectively. Let*

$n_t^{(c)}$ ($n_t^{(k)}$) *be the number of how many times t occurs in in Type(Ic) (Type(Ik)). Let*

$$m_t^{(c)} = \begin{cases} 1 \text{ if } t \in Type(I_c) \\ 0 \text{ otherwise} \end{cases} \quad m_t^{(k)} = \begin{cases} 1 \text{ if } t \in Type(I_k) \\ 0 \text{ otherwise} \end{cases}$$

be boolean functions that are true when Ic (Ik *respectively)has at least an event whose event type is* t. *Let* wt *the entropy of the event type* t. *Then:*

$$F_1(I_c, I_k) = \frac{\sum_{\forall t} \min(n_t^{(c)}, n_t^{(k)})}{\sum_{\forall t} n_t^{(c)}} \quad (3)$$

$$F_2(I_c, I_k) = \frac{\sum_{\forall t} \min(m_t^{(c)}, m_t^{(k)})}{\sum_{\forall t} m_t^{(c)}} \quad (4)$$

$$F_3(I_c, I_k) = \frac{\sum_{\forall t} w_t \cdot \min(n_t^{(c)}, n_t^{(k)})}{\sum_{\forall t} w_t \cdot n_t^{(c)}} \quad (5)$$

$$F_4(I_c, I_k) = \frac{\sum_{\forall t} w_t \cdot \min(m_t^{(c)}, m_t^{(k)})}{\sum_{\forall t} w_t \cdot m_t^{(c)}} \quad (6)$$

F1(Ic, Ik) counts how many abstract events Ic shares with Ik. This number is normalized with respect to the number of events of Ic, the current incident. On the other hand, F2(Ic, Ik) counts the number of event types shared by Ic and Ik. In other words, it considers one event per type. For example, let Type(Ic) = <A,B,B,C> the current incident and Type(Ik) = <A,B,C,D> the k − th incident in the Case Memory, where A,B,C,D are the event types of the corresponding events. If we apply F1, we obtain as result 0.75; otherwise, applying F2 we obtain 1. This is due to the fact that it counts each event type once. Notice that, F3 (F4) is similar to F1 (F2), but it also take into account the entropy of each event type. As a consequence, these functions have a discriminating power better than F1 and F2.

Definition 2 *Let Ic be the current incident, let KB={I1, . . . , Ih} be a set of past incidents, then* $Sim_x(Ic,KB) = \overline{I} \in KB$ *is the most similar past incident of Ic (according to the similarity function Fx(., .)), and it is defined as follows:*

$$F_x(I_c, \overline{I}) = \max_{I_i \in KB} F_x(I_c, I_i) \quad (7)$$

According to this definition, the procedure to find the most similar past incident is based on pattern matching. When a new attack occurs, the pattern abstracted by this attack (i.e., the abstract sequence of events) is compared with patterns in the case memory (the abstract sequence of past events) until a match is found. Consider the attack reported in Figure 2 and the case memory depicted in Figure 3. Let us suppose to use F1(., .) as similarity function, then we obtain that F1(Ic,Case1) = 0.33, F1(Ic,Case2) = 0.66, F1(Ic,Case3) = 0.33, and F1(Ic,Case4) = 0. Therefore, we select the incident Case2 as the most similar of the current one. On the other hand, applying F2(., .), we obtain F2(Ic,Case1) = 0.5, F2(Ic,Case2) = 0.5, F2(Ic,Case3) = 0.5, and F2(Ic,Case4) = 0 Finally, let us suppose that the entropy of the first and second event type of Figure 2 are 1 and 4, respectively. Applying the other retrieval functions, we obtain F3(Ic,Case1) = 0.11, F3(Ic,Case2) = 0.89, F3(Ic,Case3) = 0.11, and F3(Ic,Case4) = 0 and F4(Ic,Case1) = 0.2, F4(Ic,Case2) = 0.8, F4(Ic,Case3) = 0.2, and F4(Ic,Case4) = 0.

5 Response Adaptation, Execution and Retainment

After Response Retrieval, we have the most similar past incident of the current incident. As a consequence, we have that the response to the past incident can be used (after an appropriate adaptation) for the current incident, as well. The system uses two different plan representations: the representation of the *current response*, i.e., the representation for the plan that is developed in response to the current attack; and the representation of the *past response*, i.e., the representation for the plan stored in the case memory. A *past response* plan consists of a partially ordered sequence of *action types* (e.g., see below the response plan of incident Case2 in Figure 3).

Firewall block <IPaddress>
Send mail to <webmaster>
Update <webserver>
Firewall unblock <IPaddress>

Each action type denotes a set of actions with the same goal but that can be applied to different software platforms (i.e., actions for collecting data, for restoring normal conditions, for improving security, for communicating). Therefore, IRSS has an *action table* where for each action type we have a set of possible *concrete actions*, i.e. actions that can be automatically executed. For each concrete action, we have a set of *preconditions* hat must be true in order to apply that action. For example, the action type Firewall block hIPaddressi of the previous example has the following set of concrete actions:

```
access-list filterlist deny ip <IPaddress> any,
iptables -I FORWARD -i eth0 -s <IPaddress> -j REJECT.
```

The latter requires as precondition that the target of the attack is protected by a firewall Iptables. The *current response* plan consists of a partially ordered sequence of concrete actions, that is a script that can be immediately executed. As a consequence, the adaptation process must be accomplished in two steps. The first step has the goal of making concrete the action types. This means that for each action type in the past plan, the adaptation module looks at the action table for the corresponding set of concrete actions. For each concrete action, the module evaluates the preconditions and selects a concrete action with true preconditions. This concrete action substitutes the corresponding action type in the plan. At the end of this step, we have a plan with only concrete actions. The second step has the goal of substituting the abstract parameters with their current values. For instance, keywords like webserver in attributes source and target with current IP addresses and port numbers. The final result is a concrete plan that can be executed. In other words, a past plan represents the response strategy for all the attacks that are similar each others, the concrete plan represents the actions that must be executed to respond to the current attack. After that, the response plan is submitted to the security manager to be validated. Therefore, according to our approach, even if IRSS suggests a response plan, it will be the security manager (i.e., who has the legal responsibility of the system) to have the final decision. Notice that, this is the approach used in several diagnosis/recovery systems (e.g., in medicine [13], in fire emergency [3]). A validated plan is thus executed. After its execution, the security manager can evaluate the results and decide whether this plan must be retained. Case retainment consists of storing the abstract version of the current incident and a plan obtained substituting concrete actions and current parameters with the corresponding action types and abstract parameters. Let us consider our example, the case retrieved is Case2; this attack is a sequence of events generated when attempts are made to gain unauthorized access to a CGI application running on a WEB Server. Some applications do not perform strict checks when validating the credentials of a client host requiring the services of a server. This can lead to unauthorized access and possibly gain the privileges of the administrator. The corrective action is the response plan associated to Case2 and reported above. The first and the last actions break the connection to the attacker to limit damages. The first instruction consists of *blocking* the connection to the attacker to limit damages until the security level is improved. The connection is *unblocked* in the last action. The second instruction consists of *informing* (e.g., sending an e-mail or a SMS via the GSM system) the web master and/or all the people to which concern the incident. The third instruction consists of eliminating the vulnerability. This means installing patches or up dating the systems (e.g., operating systems, security systems, etc.).

The past plan reported above must be adapted to the current incident, therefore the adaptation module apply the previously described method. As a consequence of the fact that the web server is Apache, the firewall is Iptables, and the operating system is Linux, the adaptation module selects the concrete actions. Furthermore, the abstract parameters of the past plan must be substituted with the current parameters of the current case (e.g., the IP address 172.16.113.84:80). The result of the adaptation process is reported below.

```
iptables -I FORWARD -i eth0 -s 172.16.113.84 -j REJECT
mail > webmaster@mail.it < attackfile
apt-get update Apache
apt-get upgrade Apache
iptables -D FORWARD -i eth0 -s 172.16.113.84 -j REJECT
```

Now the security manager has to validate the plan and eventually modify it. After that he/she can execute the plan.

6. Experimental Results

We performed a set of experiments to evaluate the effectiveness of IRSS in classifying and retrieving attacks. We measured the effectiveness with two well known indices as *recall*, and *precision*. They measure the system capability of recognizing the right attacks and, thus, finding the most appropriate responses (among all the responses in the case base).

$$Recall = \frac{\|A\|}{\|B\|} \quad \textbf{(8)}$$

$$Precision = \frac{\|A\|}{\|C\|} \quad \textbf{(9)}$$

Recall is the ratio between the number of attacks that have been correctly recognized (||A||) and the number of attacks that should have been recognized (||B||). Precision is the ratio between kAk and the number of all the retrieved (||C||). We used the DARPA Data Sets [MIT Lincoln Laboratory, 1999], a standard for experimenting computer and network security tools. They consist of 125,950 log messages gathered by different kinds of sensor in two weeks, installed on a test network. During this period, the system has been attacked several times (||B|| = 53). Some of these attacks are repeated. We used these data as input of IRSS. Initially, the case memory was empty. Everytime a new attack has been detected, its description is used as input of the retrieval module. If a past case with a similarity greater than a given threshold has been retrieved, the counter of the retrieved attacks (||C||) is improved. If it has been recognized as the right attack, the counter of the recognized attack (||A||) is improved, as well. When no similar attacks were retrieved in the case memory, the current attack has been retained. We repeated these experiments for each similarity metric presented in Section 4 and with two different thresholds

(0.6 and 0.7). The results of our experiments are very promising and are reported in Table 1. We have the best results with the similarity metric F_4. We have also obtained good performances. Indeed, the fourth column of Table 1 reports the time the retrievalmodule needed for classifying all the 53 attacks. Our results are influenced by the number and the goodness of the sensors. Hence, as future work, we think to experiment IRSS with a greater number of sensors.

Similarity	Threshold	Recall	Precision	Time
F1	0.7	54.72%	93.55%	49sec
F2	0.7	77.36%	95.35%	44sec
F3	0.7	56.60%	93.75%	111sec
F4	0.7	81.13%	93.48%	95sec
…	0.6	84.90%	91.84%	

Tab. 1 Experimental results

7. Conclusions

Our work deals with a system to support the incident response activities of a security manager. Even if there exist a lot of tools to use for incident response, they usually are focused on specific activities. Which activities, when, and how we have to do is left to security manager's experience and to specific documentation about practices [5]. Furthermore, according to international security standards [8], each security manager has to define the security procedures that must be applied inside the company. Therefore, in our opinion, an automatic tool that suggests response plans depending on attacks goes towards the definition of such procedures. Obviously, these procedures must be defined according to the previous experience and must be adapted when new kinds of attacks are detected. Indeed, as can be noticed in the DARPA experiments [14], it is infrequent to have twice the same attack, but it is very common to have several "similar" attacks. These three characteristics (experience based reasoning, similarity based retrieval and learning) are typical of case based reasoning. For this reason, we based IRSS on CBR. Usually, in AI planning, there is a neat distinction between the role played by the user and the role of the planner. Indeed, usually, the user is who submits the goal to the planner, the environment sets the initial conditions, and the planner is the system that finds the solution. This approach can not be followed in the incident response domain. As a matter of fact, we must consider incident response as a joint, human and machine, planning process [3]. All these characteristics allow the system to be able to learn new responses (even for new kind of attacks) from its experience and to be improved by the evaluation of the security manager. The definition and implementation of this tool is our long term research goal. Currently, we have developed a first prototype and we have experimented the retrieval phase with the DARPA Data Set. In this paper, we report the results of these experiments. From these experiments, we have two first important results. First of all, it arises that case-based reasoning can be used to classify attacks: this is a good result. Second, we obtained an experimental evidence (see Table 1) of the intuitive notion that the most important event is that with the greatest entropy. This can be used to furtherly improve all the algorithms used in IRSS. As a future work, we have also planned to experiment the Response Adaptation and Response Retainment models, as well.

References

[1] A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[2] N. Amani, M. Fathi, and M. Dehghan.A Case-Based Reasoning Method for Alarm Filtering and Correlation in Telecommunication Networks. In *Proceedings of the Electrical and Computer Engineering*, pages 2182–2186, Canada, May 2005.

[3] P. Avesani, A. Perini, and F. Ricci. Interactive case-based planning for forest fire management. *Applied Intelligence*, 13(1):41–57, 2000.

[4] Computer Associates CA. Security Command Center, 2005. http://www3.ca.com/Solutions/ProductsAZ.aspxhttp://www3.ca.com/Solutions/ProductsAZ.aspx.

[5] CERT Coordination Center. Responding to Intrusions, 2001. http://www.cert.org/securityimprovement/modules/m06.html.

[6] M. Esmaili, R. Safavi-Naini, B. Balachandran, and J. Pieprzyk. Case-Based Reasoning for Intrusion Detection. In *Proceedings of the* 12th *Annual Computer Security Applications Conference*, pages 214–223, December 1996.

[7] A. Hätälä, C. Särs, R. Addams- Morning, and T. Virtanen. Event Data Exchange and Intrusion Alert Correlation in Heterogeneous Networks. In *Proceedings of* 8th *Colloquium for Information Systems Security Education*,West Point NY, June 2004.

[8] International Organization for Standardization. ISO17799/BS7799, 2002. http://www.iso17799software.com/http://www.iso17799software.com/.

[9] J. A. Iyer and P. Bhattacharyya. Using Semantic Information to Improve Case Retrieval in Case-Based Reasoning Systems. In *Proceedings of International Conference on the Convergence of Knowledge, Culture,*

Language and Information Technologies, Alexandria, Egypt, December 2003.

[10] J. W. Chung J. Yun, S. Ahn. Fault diagnosis and recovery scheme for web server using case-based reasoning. In *ICON '00: Proceedings of the 8th IEEE International Conference on Networks*, page 495, Washington, DC, USA, 2000. IEEE Computer Society.

[11] C. Kruegel and G. Vigna. Anomaly Detection ofWeb-based Attacks. In *Proceedings of the* 10th *ACM Conference on Computer and communications security*, pages 251–261,Washington D.C., October 2003. ACM Press.

[12] L. Lewis. A Case-Based Reasoning Approachto the Management of Faults in Communications Networks. In *Proceedings of the* 12th *Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2191–2204,March 2003.

[13] B. Lòpez and E. Plaza. Case-based planning for medical diagnosis. In *ISMIS '93: Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems*, pages 96–105, London, UK, 1993. Springer-Verlag.

[14] DARPA MIT Lincoln Laboratory. DARPA Intrusion Detection EvaluationData Sets, 1999. http://www.ll.mit.edu/IST/ideval/index.html

[15] M. Nick, B. Snoek, and T. Willrich. Supporting the IT Security of eServices with CBR-Based ExperienceManagement. In *Proceedings of* 5th *International Conference on Case-Based Reasoning Research and Development (ICCBR 2003)*, volume 2698, Trondheim Norway, June 2003.

[16] Sankar K. Pal and Simon S. K. Shiu. *Foundations of Soft Cased-Based Reasoning*. Wiley, 2004.

[17] F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transaction on Dependable and Secure Computer*, 1(3):146–169, July– September 2004.

[18] Y. S. Wu, B. Foo, Y. Mei, and S. Bagchi. Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS. In *Proceedings of the* 19th *Annual Security Applications Conference*, pages 234–244. IEEE Computer Society, 2003.

[19] E. Yilmaz, S. Stoecklin, and D. G. Schwartz. Toward a Generic Case-Based Reasoning Framework Using Adaptive Software Architectures. In *IKE*, pages 512–514, Las Vegas, Nevada, June 2003.

[20] G. Capuzzi, L. Spalazzi,F.Pagliarecci,``IRSS: An Incident Response Support System", in CTS Workshop on Collaboration and Security (COLSEC'06)}, Las Vegas, Nevada, May 14-17, USA, IEEE Computer Society Press, Los Alamitos CA, USA, 2006.

Luca Spalazzi is associate professor at the Università Polietcnica delle Marche. He received the M.S. in Electronic Engineering and the Ph.D in Artificial Intelligent Systems from the University of Ancona, Italy in 1989 and 1994, respectively. He has worked as consultant at the Istituto di Ricerca Scientifica e Tecnologica (IRST), Trento, Italy. He was a visiting scholar at the Australian Artificial Intelligence Institute (AAII), Carlton, Vic., Australia and at the Computer Science Department, Stanford University, California. His present research areas include Computer and Network Security, Case-based Reasoning, and Multi-agent Systems.

Gianluca Capuzzi is a Ph.D student at the Università Politecnica delle Marche. He received the M.S. In Electronic Engineering from the Università Politecnica delle Marche in 2004. His reserach areas include Computer and Network Security and Case-based Reasoning. He was a visiting scholar at the SRI International, Menlo Park, California.

Egidio Cardinale is a Ph.D student at the Università Politecnica delle Marche. He received the M.S. In Electronic Engineering from the Università Politecnica delle Marche in 2005. His reserach areas include Computer and Network Security and Case-based Reasoning.

Ivan Di Pietro is a Ph.D student at the Università Politecnica delle Marche. He received the M.S. and the B.S. in Computer Engineering from the Università Politecnica delle Marche, in 2006 and 2004, respectively. His reserach areas include Computer and Network Security and Case-based Reasoning.