Generation of Attack Scenarios for Evaluating IDS

Mohammed Saber, Toumi Bouchentouf, and Abdelhamid Benazzi

***Abstract*—We focus in this paper to improve the level of intrusion detection system (IDS). This improvement is based on three research areas: classification of attacks, generation of attack scenarios and finally evaluation methods. We will discuss in this article the second area, which consists on the research of meaningful scenarios in order to minimize false and positive alerts reported by an IDS. We will present two algorithms generating these scenarios. The first one allows the conversion of the problem to a constraint programming problem (CSP) and the second one is based on an algorithm to search the shortest path. We will also compare the results of these two algorithms.**

***Index Terms*—Scenario, attack, evaluation, IDS, CSP, CHOCO.**

I. Introduction

Our main research area is to test and evaluate IDS (Intrusion Detection Systems). The objective is to develop a classification model of attacks (class model of attacks) then model the attack process and generate attack scenarios.

Regarding the classification model attacks in Saber and al. [1, 2] we have presented a better classification model attacks Gad and al. [3] by eliminating duplication of classes. This allowed us to reduce the number of meaningful classes by using the method CTM (Classification Tree Method) [4] using the tool CTE (Classification Tree Editor) [5].

The purpose of classification is to minimize false and positive alerts reported by IDS. But attacks follow several scenarios depending on the nature and purpose of the attack which makes the implementation of these different classifications very hard on IDS.

In this paper, we focus on the generation of attack scenarios. We adopted the process model of malware attacks Gad and al. [6]. We propose two algorithms for generating attack scenarios from this model. Our goal is to generate a significant minimum number of attack scenarios in a minimal time, which will facilitate the integration model in an IDS, and thus facilitate its evaluation. Indeed, the aim is that the IDS can detect an attack as quickly as possible before it becomes an intrusion.

This paper is composed as follows: in section two we will make a description of the attack process model of Gad and al. [6]. We will detail in section three the modeling problem, we will present the two algorithms used to generate attack scenarios in Section four. In the fifth section we will present the results and then we will start a discussion. We will end with a conclusion and future works.

Manuscript received April 9, 2012; revised May 1, 2012.

M. Saber and T. Bouchentouf are with the Department of Computer Science, National School of Applied Sciences, Mohammed First University, Oujda, Morocco (e-mail: mosaber@gmail.com, tbouchentouf@gmail.com).

A. Benazzi is with the Department of Computer Science, High School of Technology, Mohammed First University, Oujda, Morocco (e-mail: benazzihamid@yahoo.fr).

II. Model of Attacks Process

There are several models of attacks [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. They are generally specific to the runtime environment, and therefore require a precise and detailed knowledge of the architecture, topology and network vulnerabilities and considered system. Moreover, these models are based primarily on known vulnerabilities and ignore the attacks that may exploit still unknown vulnerabilities, which would constitute a serious limitation, since the robustness of IDS depends also on unknown vulnerabilities and new attacks.

In this paper we have adopted the model of the attack process of Gad and al. [6] which is based on a preliminary analysis of malware attacks like the most prevalent viruses and worms. This choice is justified by the fact that this model is the result of the analysis of more than 70 malware from the CME List (Mitre's Common Malware Enumeration list) [18], which are representative of the most dangerous and more widespread attacks. Indeed, given that worms are autonomous, they must include all the steps in an attack process. In addition, viruses such as worms can be seen as a class of automated attacks developed by skilled attackers, and this can help to understand how interactive attacks can be conducted.

This model is described in Figure (Fig. 1). It distinguishes the following steps:

1) **Recognition (Reconnaissance):** it is logical for an attacker to find the necessary information on potential victims before targeting them with the most appropriate attack tools (exploit codes, toolkits).
2) **Gain access:** to achieve their objectives, attackers usually need access to victims resources, the level of access required will obviously depend on the attack. However, some types of attacks such as denial of service attacks do not need access to the victim machine.
3) **Privilege Escalation:** Access originally obtained by the attacker is sometimes insufficient to achieve the attack, in which case, the attacker tries to increase its privileges to have more power (for example, switch from user mode to administrator mode to access the system resources).
4) **Browsing Victim:** after having acquired sufficient privileges, the attacker usually tries to explore the machine or the target network (eg, searching files and directories), to search for a particular account (as a guest account or an anonymous ftp account), to identify the hardware components, to identify installed programs or to search for trusted hosts (typically, those with certificates installed on the victim machine).

5) **Principal Actions:** as shown in Figure (Fig. 1), this step may take different forms, for example, an attacker can execute a denial of service attacks, install malicious code, compromising the integrity of data or run a program.
6) **Hiding Traces:** the most experienced attackers generally use this last step to erase their tracks, thereby making detection more difficult.

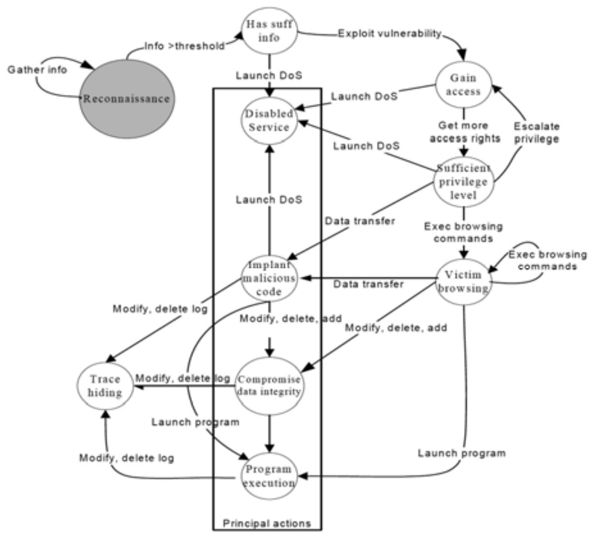

Fig. 1. Model of attack process.

To generate attack scenarios, Gad and al. proposed a simplified model called the state machine, illustrated in Figure (Fig. 2). The steps taken by the malware attacks can be classified into only 8 primitives and each identified by a symbol, as indicated below:

1) **R:** Recognition (Reconnaissance)
2) **VB:** Exploration of the machine /or the network of the victim (Victim Browsing)
3) **EP:** Program execution (Execute Program)
4) **GA:** Gain Access
5) **IMC:** Implementation of malicious code (Implant Malicious Code)
6) **CDI:** Compromise of integrity (Compromise Data Integrity)
7) **DoS:** DoS (Denial of Service)
8) **HT:** erasing traces (Hide Traces)

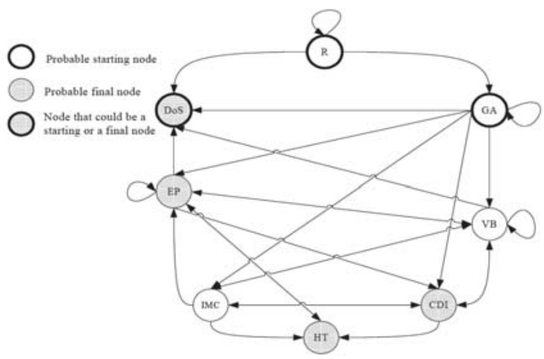

Fig. 2. State machine representing the attack process.

The graph in Figure (Fig. 2) allows the generation of attack scenarios at an abstract level. By applying constraints on the paths between nodes and consecutive repetition of the same action (loops), as shown in the connection matrix of Figure (Fig. 3), we can find a set of valid abstract scenarios. A valid scenario is a combination of these nodes with constraints that lead to an attack or intrusion. Here are some examples of scenarios that we can generate from the graph in Figure (Fig. 2):

1) Scén_1 = (*R*, GA, DoS): Begins recognizing then obtaining gain access and end with a DoS attack.
2) Scén_2 = (*R*, DoS): Begins recognizing and ends with a DoS attack.
3) Scén_3 = (*R*, *R*, *R*, DoS): Begins recognizing several times and end with a DoS attack, this scenario is equivalent to the scenario Scén_2 = (*R*, back).

	R	GA	DoS	VB	CDI	EP	IMC	HT
R	1	1	1	0	0	0	0	0
GA	0	1	1	1	1	1	1	0
DoS	0	0	0	0	0	0	0	0
VB	0	0	1	1	1	1	1	0
CDI	0	0	0	1	1	1	1	1
EP	0	0	1	1	1	0	1	1
IMC	0	0	0	1	1	1	0	1
HT	0	0	0	0	0	1	0	0

Fig. 3. Connection matrix.

1) **1:** there is a relationship between two nodes; a node is the son of another node.
2) **0:** no relationship between two nodes, one node is not the son of another node.

It is important to note that this iterative approach for generating attack scenarios has overcome the problem of combinatorial explosion, inherent problem to conventional approaches to generating attack scenarios.

The problem we want to solve is to find efficient algorithms that can generate valid meaningful attack scenarios. It would be easier to incorporate the state machine model in an IDS, to test and evaluate it. The modeling of our problem is presented in the next section.

III. Modeling The Problem

In order to facilitate the modeling we will use Figure (Fig. 4), And the Correspondence table (TABLE I):

TABLE I: Matching State Table.

Node	Corresponding value	Node	Corresponding value
R	1	EP	5
GA	2	IMC	6
DoS	3	HT	7
VB	4	CDI	8

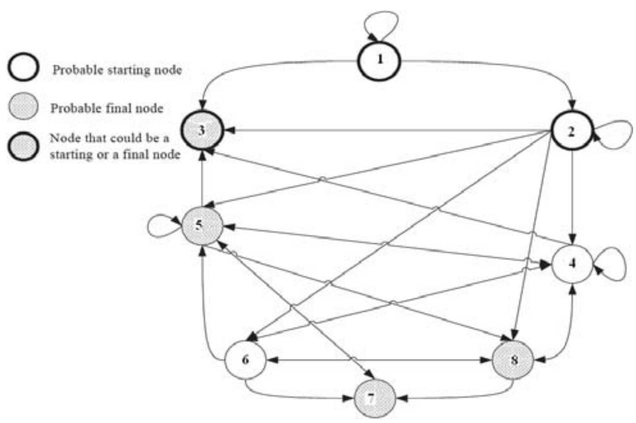

Fig. 4. Matching state machine.

ND is the set of all nodes : ND={1,2,3,4,5,6,7,8}
$ND_{departure}$ is the set of starting nodes $ND_{departure}$ = {1,2,3}
ND_{final} is the set of final nodes ND_{final} = {3,4,7,8}
L (ND) is the set of the subset of ND, i.e

$$X \in L(ND) \Leftrightarrow X \subset ND$$

R is the relation defined by:

$$ND \to L(ND)$$
$$R : x \to R(x) = X$$

X is the set of the nodes son of x.
Example : R(1)={1,2,3}; R(2)={2,3,4,5,6,8}; R(3)= ∅// empty set; R(4)={3,4,5,6}; R(5)={3,4,5,7}; R(6)={5,7}; R(7)= ∅// empty set; R(8)={4,6,7}.

A. Definition 1: Scenario

A scenario SN_k of size K is k-uple (x_1, x_2, x_3, …, x_k) such as: x_i is the son of node $x_{i\text{-}1}$, one notes:

$$SN_k = (x_1, x_2, ..., x_k) \text{ with } x_i \in R(x_{i-1}) \forall i \neq 1$$

For example: $(x_1 = R) \Rightarrow (x_2 \Rightarrow GA) \Rightarrow (x_3 \Rightarrow DoS)$

B. Definition 2: Valid Scenario

SN_k is a valid scenario if and only if:

$$SN_k \text{ is a scenario}$$
$$x_1 \in ND_{departure}$$
$$x_k \in ND_{final}$$

SVk is the set of valid scenarios having k nodes:

We have SVk = {SNk / SNk is a valid scenario of size k}

cardND = card(ND) : the number of the ND elements.

SV is the set of all valid scenarios:

$$SV = \bigcup SV_k \text{ or } 1 \leq k \leq +\infty$$

C. Definition 3: Equivalent Scenarios

Two scenarios A and B are equivalent if and only if:

$$\exists p \in ND^n \text{ and } \exists q \in ND^m \text{ with } (n, m) \in \mathrm{N} \times \mathrm{N}$$

Such as:

$A = (p, p, q)$ and $B = (p, q)$ or $A = (q, q, p)$ and $B = (q, p)$

SE_k is the set of the valid equivalent scenarios, we notes:

SE_k = {SV_k / SV_k is a valid scenario of size k}

SE is the set of valid equivalent scenarios, given:

$$SE = \bigcup SE_k \text{ or } 1 \leq k \leq cardND$$

D. Particular Case K=1

SN_1 = (x_1) is valid if and only if:

$$X_1 \in ND_{departure} \cap ND_{final} \text{ thus } x_1 \; ND_{departure} \cap ND_{final}$$

We propose in the next section to presentation of the two algorithms for generation of attack scenarios. The first algorithm reduces the generation of scenarios to solve a problem of constraint programming (CSP) and the second algorithm is based on a modified algorithm shortest path (MASP).

IV. Presentation of The Tow Algorithms

A. Presentation of CSP (CSPA) and CHOCO

A CSP "constraint satisfaction problem" is modeled as a set of constraints imposed on variables, each of these variables taking values in a domain. More formally, a CSP will be defined by a triplet (V, D, C) such that:

1) The set of variables (unknowns) of the problem is:

$$V = \{x_1, x_2, ..., x_k\}$$

2) D is the function that maps each variable x_i to its domain $D(x_i)$, wich means all possible values of x_i.

$$x_1 \in ND_{depature}$$
$$x_k \in ND_{final}$$
$$D(x_i) = \bigcup R(y) \forall y \in D(x_{i-1}) \text{ or } i \neq 1 \text{ and } i \neq k$$

3) C = {c_i / x_i ∈ $R(x_{i\text{-}1})$ or ∀ i≠1 and i≠k} is the set of constraints. Each constraint is a relation between certain variables V, restricting the values that these variables can take simultaneously.

The implementation of solving algorithms for this problem uses several languages such as Mozart [19], Jacop [20], ILOG [21] and the java library of Choco solver [22]. We have adopted the CHOCO library to generate our algorithm.

The resolution of the CSP thus formulated, has been implemented in Java using the API (Application Programming Interface) CHOCO library. This solver requires the submission of a description of the variables, their domains and the set of constraints as shown in Figure (Fig. 5).

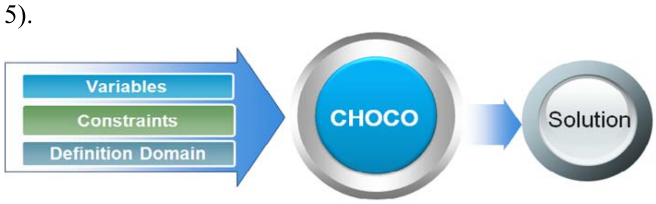

Fig. 5. CHOCO Operating principle.

B. Modified Algorithm Shortest Path (MASP)

Principle of the algorithm:

1) **Input:** node name of a departure "departure" (list of visited nodes "ListVisitedNode" used to avoid infinite recursion (infinite loop), this list is initially empty.
2) **Output:** list of valid scenarios for this departure.

Begin algorithm:
list ListDepartureNodes ← {"R","GA","DoS"};
list ListValidScenarios ← empty;
list AValidScenario ← empty;
For each element "departure" from the list "ListDepartureNodes" do:
list ListVisitedNodes ← empty;

```
  list
ListValidScenarios←GetAchievinObjectiveBy(departure,
ListVisitedNodes);
        For each scenario "AValidScenario" from the list
        "ListValidScenarios" do:
                Print (AValidScenario);
        End for
End for
Return ListValidScenarios ;
End algorithm.
```

GetAchievinObjectiveBy function is defined as follows:

```
Function GetAchievinObjectiveBy (departure,
ListVisitedNodes)
  ListVisitedNodes.add(departure);
  For each node "SonNode" directly reaches the node
   "departure" do:
     If  ListVisitedNodes does not SonNode then:
        list ListSemiScenario ←GetAchievinObjectiveBy
              (SonNode, ListVisitedNodes);
           For each scenario "SemiScenario" from the list
           "ListSemiScenario" do:
                ListFinalSenario.add (departure + "==>" +
                SemiScenario);
     End For
       End If
   End For
        If the node "departure" is a final node then:
    ListSenarioFinal.add (departure);
  End If
Return ListFinalSenario;
Fin Function
```

V. Obtained Results and Discussion

We implemented both algorithms CSPA and MASP by developing an application DIA (Detector Intrusion Automatic) (Fig. 6) using the framework Zk [23]. This application allows data acquisition and results display. The algorithm CSPA was implemented using the CHOCO library and Java language and the algorithm MASP was implemented in Java language. Indicating the size of the desired valid scenario, we get the number of valid scenarios and the time taken in search by the two algorithms implemented. So we have calculated the number of valid scenarios of size k, k varying from 1 to 8, and the time needed to find a scenario of size k. The following table summarizes the results.

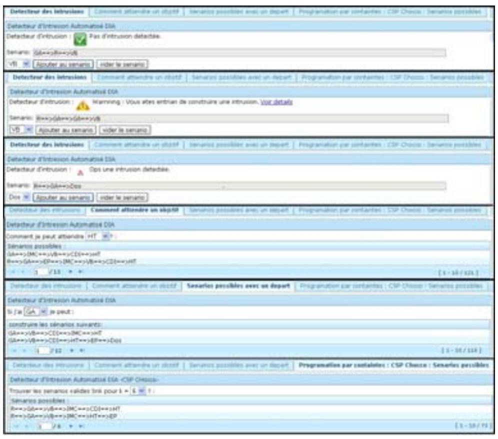

Fig. 6. DIA: Detector intrusion automatic.

TABLE II: Obtained Results.

Scenario size k	Number of valid scenarios	CSPA Time needed in milliseconds (ms)	MASP Time needed in nanoseconds (ns)
1	1	1	5
2	4	10	80
3	13	15	150
4	34	24	200
5	63	49	270
6	73	68	410
7	42	63	450
8	8	213	600

Scenarios thus generated are 238 valid scenarios for the two algorithms, and required an average of 405 milliseconds for CSPA and 2 milliseconds for MASP. We also found in the two algorithms that the largest the scenario is, the more time the generation of valid scenarios takes (Fig. 7).

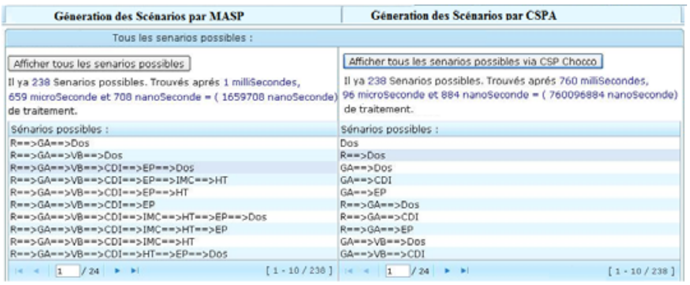

Fig. 7. Senarios MASP and CSPA.

We infer that the MASP algorithm is better than the CSPA algorithm in terms of performance per cons it requires more resources because the data are stored on the RAM (Random Access Memory), while the CSPA algorithm makes all the combinations according to the size k and then extracts the valid scenarios within the constraints (Fig. 8).

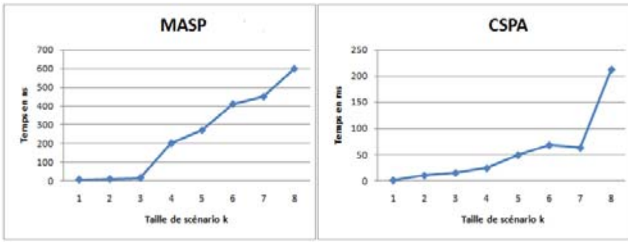

Fig. 8. Time MASP and CSPA.

So that the two solutions can be implemented in an IDS, it is first necessary to implement the model cited in Figure (Fig. 1) and then evaluate it in relation to other IDS that deal malware attacks like antivirus.

VI. Conclusion and Future Works

In this paper we presented an algorithm based on CSP to generate meaningful attack scenarios based on the model proposed by Gad and al. [6] to represent attacks like malware (viruses, Trojan.). The first is named CSPA, it is based on constraint programming by modeling the problem as a CSP problem, and the second one is MASP, it is based on a search algorithm of the shortest path. We implemented the CSPA through the CHOCO library and Java language and the MASP in Java language.

We then developed an application using ZK framework to compare the two algorithms. So we have calculated the number of valid scenarios of size k, k varying from 1 to 8, 8 the maximum number of nodes in the model proposed by GAD for malware, and the time needed to find a scenario of size k.

We deduced that MASP algorithm is better than CSPA algorithm in terms of performance per cons it requires more resources.

A very interesting perspective would be to take inspiration from the two algorithms and apply them on other models of attacks. It is also interesting to look for other algorithms to be able to find the best performing one. Another perspective is to implement an IDS prototype implementing the GAD model for malware and integrating these two algorithms so that we can compare the prototype such built with other IDS like antivirus.

References

[1] M. Saber, T. Bouchentouf, A. Benazzi, and M. Aziz "Amelioration of Attack Classifications for Evaluating and Testing Intrusion Detection System," *Journal of Computer Science* vol. 6, no. 7, pp. 716-722 , 2010

[2] M. Saber, T. Bouchentouf, A. Benazzi, and M. Azizi, "Attacks classification for evaluating intrusion detection system," *Proceeding of IADIS International Conferences Informatics 2010, Wireless Applications and Computing 2010 and Telecommunications, Networks and Systems 2010*, pp. 166 – 170.

[3] Mohammed S. Gadelrab, Anas Abou El Kalam, and Yves Deswarte, "*Defining categories to select representative attack test-cases*," *Proceedings of the 2007 ACM workshop on Quality of protection (QoP '07)*, Alexandria, Virginia, USA, pp. 40-42, 2007.

[4] Classification Tree Method 2010 : http://www.systematic-testing.com.

[5] Classification Tree Editor 2010 : http://www.systematic-testing.com.

[6] Mohammed S. Gadelrab, Anas Abou El Kalam, and Yves Deswarte, "Execution Patterns in Automatic Malware and Human-centric Attacks," *NCA 2008: Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications* vol. 29-36.

[7] L. Chen, "Modeling Distributed Denial of Service Attacks and Defenses," PhD thesis, Carnegie Mellon University, USA, 2003.

[8] S. Cheung, U. Lindqvist, and M. Fong, "Modeling Multistep Cyber Attacks for Scenario Recognition," *Proceedings of the Third DARPA Inofrmation Survivability Conference and Exposition (DISCEX III),* Washington, DC, USA, pp. 284-292, 2003.

[9] A. Garg, S. Upadhyaya, and K. Kwiat, "Attack Simulation Management for Measuring Detection Model Effectiveness," *Proceedings of The Second Secure Knowledge Management Workshop (SKM 2006),* Brooklyn, NY, USA, 2006.

[10] T. Tidwell, R. Larson, K. Fitch, and J. Hall, "Modeling Internet Attacks," *Proceeding of the IEEE Workshop on Information Assurance and Security, West point*, NY, USA, pp. 54-59, 2001.

[11] B. Schneier, "Attack Trees: Modeling Security Threats," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21-29, 1999.

[12] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," *Proceeding of 2002 IEEE Symposium on Security and Privacy, Oakland,* California, USA, pp. 273-284, 2002.

[13] J. Steven Templeton, and Karl Levitt, "A requires/provides model for computer attacks," *Proceedings of the 2000 workshop on New security paradigms (NSPW '00),* NY, USA, pp. 31- 38, 2000.

[14] J. P. McDermott, "Attack net penetration testing," *Proceedings of the 2000 workshop on New security paradigms (NSPW '00),* New York, USA, pp. 15-21, 2000.

[15] Ole Martin Dahl and Stephen D. Wolthusen, "Modeling and Execution of Complex Attack Scenarios using Interval Timed Colored Petri Nets," *Proceedings of the Fourth IEEE International Workshop on Information Assurance (IWIA '06),* Washington DC, USA, pp. 157-168, 2006.

[16] Marc Dacier and Yves Deswarte, "Privilege Graph: an Extension to the Typed Access Matrix Model," *Proceedings of the Third European Symposium on Research in Computer Security(ESORICS '94),* London, UK, pp. 319-334, 1994.

[17] Mohamed Kaaniche, Y. Deswarte, Eric Alata, Marc Dacier, and Vincent Nicomette, "Empirical analysis and statistical modeling of attack processes based on honeypots," *Proceeding of Workshop on Empirical Evaluation of Dependability and Security (WEEDS), DSN'2006,*Philadelphia, USA, pp. 119-124, 2006.

[18] Mitres Common Malware Enumeration list http://cme.mitre.org/

[19] The Mozart Programming System. [Online]. Available: http://www.mozart-oz.org.

[20] JaCoP. [Online]. Available: http://jacop.cs.lth.se/

[21] IBM. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/

[22] HOCO. [Online]. Available: http://www.emn.fr/z-info/choco-solver/

[23] ZKOSS. [Online]. Available: http://www.zkoss.org/

Mohammed Saber Engineer in networks and system at the Computer Science Department, National School of Applied Sciences, Mohammed First University Oujda (Morocco), and doctoral researcher at the MATSI laboratory (Applied Mathematics, Signal Processing and Computer Science Laboratory) in the field of computer security. I am interested in intrusion detection system and attacks.

Toumi Bouchentouf Professor in computer science, responsible for the computer engineering department. From the National School of Applied Sciences, Mohammed First University Oujda (Morocco). Interested in software engineering, management and monitoring of IT project, computer security in intrusion detection system and attacks, and multi-agent system.