Improved Off-Line Intrusion Detection Using A Genetic Algorithm And RMI

Ahmed AHMIM[1]
Department of Computer Science,
Badji Mokhtar University
Annaba, 23000, Algeria
ahmed.ahmim@hotmail.fr

Nacira GHOUALMI[2]
Department of Computer Science,
Badji Mokhtar University
Annaba, 23000, Algeria
ghoualmi@yahoo.fr

Noujoud KAHYA[3]
Department of Computer Science,
Badji Mokhtar University
Annaba, 23000, Algeria
kahya.noudjoud@gmail.com

***Abstract*— This article proposes an optimization of using Genetic Algorithms for the Security Audit Trail Analysis Problem, which was proposed by L. Mé in 1995 and improved by Pedro A. Diaz-Gomez and Dean F. Hougen in 2005. This optimization consists in filtering the attacks. So, we classify attacks in "Certainly not existing attacks class", "Certainly existing attacks class" and "Uncertainly existing attacks class". The proposed idea is to divide the 3rd class to independent sub-problems easier to solve. We use also the remote method invocation (RMI) to reduce resolution time. The results are very significant: 0% false+, 0%false-, detection rate equal to 100%. We present also, a comparative study to confirm the given improvement.**

Keywords-component; intrusion detection system; Genetic Algorithm; Off-Line Intrusion Detection; Misuse Detection;

I. INTRODUCTION

The computing networks became the paramount tool for the various sectors (social, economies, military… etc.). The phenomenal developments of networks are naturally accompanied by the increase in the number of users. These users, known or not, are not necessarily full of good intentions for these networks. They can exploit the vulnerabilities of networks and systems, to try access to sensitive information in order to read, modify or destroy them. Therefore, that these networks appear the targets of potential attacks, their securing has become an unavoidable bet.

Computer security has become in recent years a crucial problem. It rallies the methods, techniques and tools used to protect systems, data and services against the accidental or intentional threats, for ensure: Confidentiality; Availability; Integrity [1].

Nowadays, different techniques and methods have been developed to implement a security policy: authentication, cryptography, firewalls, proxies, antivirus, Virtual Private Network (VPN), Intrusion Detection System (IDS). This paper is organized after an introduction as: The second section is a state of the art on the IDS. The third section presents a formalization of the Security Audit Trail Analysis Problem (SATAP) as well as using Genetic Algorithms for the Security Audit Trail Analysis Problem proposed by Mé [2]; the fourth section presents our contribution to optimize using Genetic Algorithms for the Security Audit Trail Analysis Problem. The fifth section presents the results obtained by our approach; the sixth section presents a comparative study between the two approaches. Finally, the conclusion presents the advantages of our approach, and the prospects work.

II. THE INTRUSION DETECTION SYSTEMS

Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems [3]. The intrusion detection system was introduced by James Anderson [4], but the subject didn't have great success. After that, Denning defined the intrusion detection system models [5], where he exhibits the importance of security audit, with the aim to detect the possible violations of system security policy.

According to Intrusion Detection Working Group of IETF an intrusion detection system includes three vital functional elements: information source, analysis engine, and response component [6].

There are five concepts to classify intrusion detection Systems, which are: The detection method; The behavior on detection; The audit source location; The detection paradigm; The usage frequency [6].

The detection method is one of the principal characters of classification they describe the characteristics of the analyzer. When the intrusion detection system uses information about the normal behavior of the system it monitors, we qualify it as behavior-based. When the intrusion detection system uses information about the attacks, we qualify it as knowledge-based [6].

III. INTRUSION DETECTION BY SECURITY AUDIT TRAIL ANALYSIS

The Security Audit is as medical diagnosis, in order to determine the set of conditions, which may explain the presence of observed symptoms (in IDS: the recorded events in the audit trail). For this reason, expert uses specific knowledge (the scenarios of attack) based cause at an effect.

The expert uses its knowledge to develop assumptions that confront the reality observed. If there are still observed symptoms than the made hypothesis made is wrong. On the other hand, if there are more symptoms than those observed in the reality, a new hypothesis more relevant must be tested [2].

In this approach, the attack scenarios are modeled as a set of couples (E_i, N_i) where E_i is the type of event and N_i is the number of occurrences of this type of event in the scenario. This approach is called « the Security Audit Trail Analysis Problem».

A. Specification of the Security Audit Trail Analysis Problem [7]

Formally, the Security Audit Trail Analysis Problem can be expressed by the following statement:

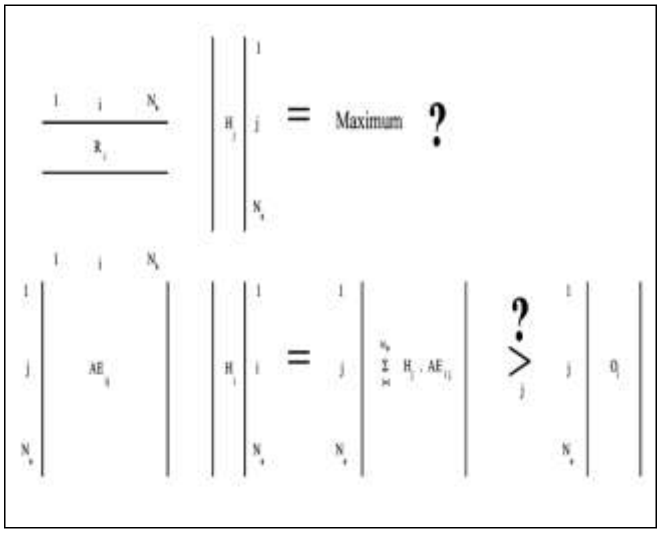


Figure 1. The Security Audit Trail Analysis Problem [7]

- $\boldsymbol{N_e}$: the number of type of audit events.
- $\boldsymbol{N_a}$: the number of potential known attacks.
- $\boldsymbol{AE}$: is the $N_a \times N_e$ attacks-events matrix which gives the set of events generated by each attack. AE_{ij} is the number of audit events of type i generated by the scenario j

— $AE_{ij} \geq 0$ (1)

- $\boldsymbol{R}$: is $N_a$dimensional weight vector, where:

— $(R_i > 0)$ (2)

— is the weight associated to the attack i (R_i is proportional to the risk inherent in the attack scenario i).

- $\boldsymbol{O}$: is the $N_e$dimensional vector where:

— O_i counts the occurrence of events of type i present in the audit trail (O is "observed audit vector").

- $\boldsymbol{H}$: is $N_a$dimensional hypothesis vector, where:

$H_i = 1$ (3)

(a) If the attack i is present according to the hypothesis and

$H_i = 0$ (4)

(b) Otherwise (H describes a particular attack subset).

(c)

To explain the data contained in the audit trail (i.e. O) by the occurrence of one or more attack. We have to find the H vector which maximizes the $R \times H$ Product (it's the pessimistic approach: finding H so that the risk is the greatest) with the constraint:

$(AE \times H)_i \leq O_i, (1 \leq i \leq N_a)$ (5)

Finding H vector is NP-complete. Consequently, the application of classical algorithms is therefore, impossible where $N_a$equals to several hundreds.

The heuristic approach that we have chosen to solve that NP-complete problem is the following: a hypothesis is made (e.g. among the set of possible attacks, attacks i, j and k are present in the trail), the realism of the hypothesis is evaluated and, according to this evaluation, an improved hypothesis is tried, until a solution is found.

In order to evaluate a hypothesis corresponding to a particular subset of present attack, we count the occurrence of events of each type generated by all the attacks of the hypothesis. If these numbers are less than or equal to the number of events recorded in the trail, then the hypothesis is realistic.

After, we have to find an algorithm to derive a new hypothesis based on the past hypothesis: it is the role of the genetic algorithm.

B. Using Genetic Algorithms for Misuse Detection [7]

Genetic algorithms (GA) are optimum search algorithms based on the mechanism of natural selection in a population. A population is a set of artificial creatures (individuals or chromosomes). These creatures are strings of length 1 coding a potential solution to the problem to be solved, most often with a binary alphabet. The size L of the population is constant. The population is nothing but a set of points in a search space. The population is randomly generated and then evolves in every generation. A new set of artificial creatures is created using the fittest or pieces of the fittest individuals of the previous one. The fitness of everyone is simply the value of the function to be optimized (the fitness function) for the point corresponding to the individual. The iterative process of population creation is achieved by three basic genetic operators: selection (selects the fittest individuals), reproduction or crossover (promotes exploration of new regions of the search space by crossing over parts of individuals) and mutation (protects the population against an irrecoverable loss of information).

Two challenges arise when applying GAs to a particular problem: coding a solution for that problem with a string of bits and finding a fitness function to evaluate everyone of the population.

1) Coding a Solution with a Binary String [7]

An individual is a one length string coding a potential solution to the problem to be solved. In our case, the coding is straightforward: the length of an individual is $\mathrm{N_a}$ and each individual in the population corresponds to a particular H vector.

2) The Fitness Function [7]

We have to search, among ail the possible attack subsets, for the one which presents the greatest risk to the system. This result in the maximization of the product $\mathrm{R} \times \mathrm{H}$. As GAs are optimum search algorithms, finding the maximum of a fitness function, we can easily conclude that in our case this function should be made equal to the product $\mathrm{R} \times \mathrm{H}$. So we have:

$$F = \sum_{I=1}^{Na} R_i \, I_i \tag{6}$$

Where I is an individual.

This fitness function does not take into account the constraint feature of our problem, which implies that some individuals among the $2^{\mathrm{N_a}}$ Possible are not realistic.

This is the case for some i type of events when:

$$(AE \ \times H) \ _i > \ O_i \tag{7}$$

As a large number of individuals do not respect the constraint. We decided to penalize them by reducing their fitness values. So we compute a penalty function (P) which increases as the realism of this individual decreases: let $\mathrm{T_e}$ be the number of types of events for which

$$(\mathrm{AE} \ \times \mathrm{H}) \ _\mathrm{i} > \ \mathrm{O_i} \tag{8}$$

The penalty function applied to such an H individual is then:

$$\mathrm{P} = \ \mathrm{T_e^p} \tag{9}$$

A quadratic penalty function (i.e. p = 2) allows a good discrimination among the individuals. The proposed fitness function is thus the following:

$$\mathrm{F(I)} = \alpha + \left(\sum_{\mathrm{i=1}}^{\mathrm{Na}} \mathrm{R_i \, I_i} - \beta . \mathrm{T_e^p}\right) \tag{10}$$

The β parameter makes it possible to modify the slope of the penalty function and α sets a threshold making the fitness positive. If a negative fitness value is found, it is equaled to 0 and the corresponding individual cannot be selected. So the parameter allows the elimination of a too unrealistic hypothesis.

This selective function was improved by Diaz-Gomez, P. A. Hougen [8]. This improvement proved mathematically [9] [10]. A new selective function provides less false positives and less false negative [11].

The new selective function is:

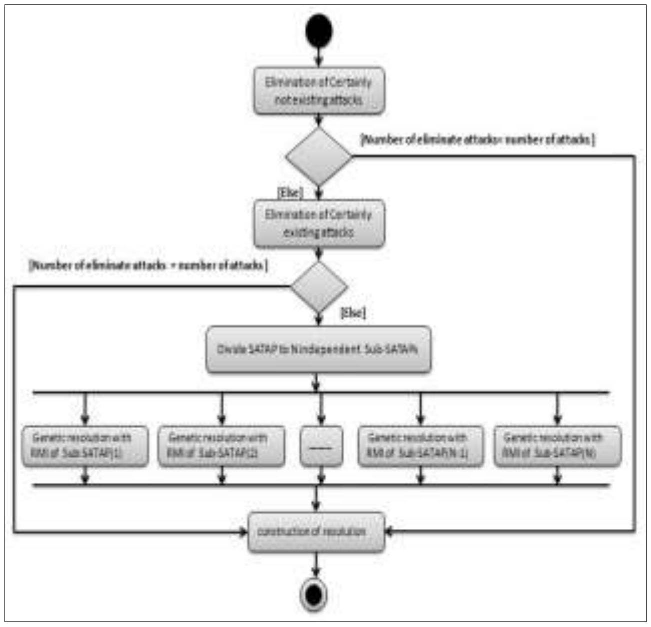
$$\mathrm{F(I)} = \mathrm{N_e} - \mathrm{T'} \tag{11}$$

Where $\mathrm{N_e}$ corresponds to the total number of classified events. $\mathrm{T'}$Corresponds to the number of overestimates, i.e., the number of times $(\mathrm{AE} \ \times \mathrm{H}) \ _\mathrm{i} > \ \mathrm{O_i}$ for each attack H_i. That is, if a hypothesized attack H_i considered alone, would cause $(\mathrm{AE} \ \times \mathrm{H}) \ _\mathrm{i} > \ \mathrm{O_i}$ for some i, and another hypothesized attack H_j considered alone, would also cause $(\mathrm{AE} \ \times \mathrm{H}) \ _\mathrm{i} > \ \mathrm{O_i}$, then $\mathrm{T'}$ would have a value of 2 [12].

IV. CONTRIBUTIONS

Inspired from Ludovic Mé [7] contributions and Diaz-Gomez, P. A. Hougen [8] improvement, we classify attacks in Security Audit Trail in three classes and divide the 3rd class to independent sub-problems. Then, we apply the genetic algorithm with the proposed crossover operator in [13] and L. Mé selective function (10). The second contribution is to optimize the resolution time of the genetic algorithm. For this we apply RMI (remote method invocation) to each sub-problem.

The Figure2 represents the activity diagram that summarizes the different steps of our proposition.


Elimination of Certainly not existing attacks
[Number of eliminate attacks= number of attacks]
[Else]
Elimination of Certainly existing attacks
[Number of eliminate attacks = number of attacks]
[Else]
Divide SATAP to N independent Sub-SATAPs
Genetic resolution with RMI of Sub-SATAP(1)
Genetic resolution with RMI of Sub-SATAP(2)

Genetic resolution with RMI of Sub-SATAP(N-1)
Genetic resolution with RMI of Sub-SATAP(N)
construction of resolution

Figure 2. Activity diagram of our contribution

A. *Filtration of attacks*

The Filter uses Observation matrix "O" and the Matrix attack-event "AE'. The proposed idea is reducing the size of the problem in order to obtain the correct solution and to reduce the runtime.

Consequently, we classify attacks in three classes, and divide the last class to sub-problems:

1) *Certainly not existing attacks' class*

Eliminate attacks, which have a probability of existence equal to 0%. These attacks generate an occurrence number for one of the events greater than the occurrences number audited for this event. So, attacks i satisfy the following formula:

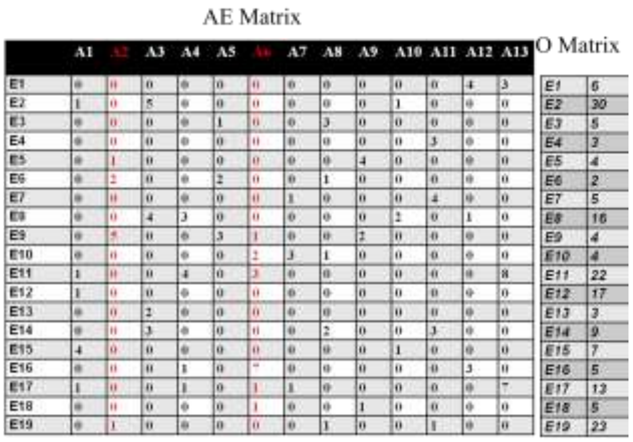
$$\exists\, j \in N_e \ (AE_{ij} > O_j) \qquad (12)$$

To eliminate these attacks, we compare Matrix attack-event "AE" with the Observation matrix "O". The result is attacks noted $\mathrm{N_{ap}}$. After that, we remove the events that have value 0 in Observation matrix "O". The number of events used for selected attacks is noted $\mathrm{N_{ep}}$.

The results are matrixes with the following dimensions:

$$AE_{(N_{ap},N_{ep})}\ ,O_{(N_{ep})}\ ,H_{(N_{ap})}\ ,R_{(N_{ap})} \qquad (13)$$

Figure 3 shows an example of elimination certainly not existing attacks. For example, attack A2 generates 5 events E9, while the Security Audit Trail records only 4 events E9.


AE Matrix

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
E1	0	0	0	0	0	0	0	0	0	0	0	4	3
E2	1	0	5	0	0	0	0	0	0	1	0	0	0
E3	0	0	0	0	1	0	0	3	0	0	0	0	0
E4	0	0	0	0	0	0	0	0	0	0	3	0	0
E5	0	1	0	0	0	0	0	0	4	0	0	0	0
E6	0	2	0	0	2	0	0	1	0	0	0	0	0
E7	0	0	0	0	0	0	1	0	0	0	4	0	0
E8	0	0	4	3	0	0	0	0	0	2	0	1	0
E9	0	5	0	0	3	1	0	0	2	0	0	0	0
E10	0	0	0	0	0	2	3	1	0	0	0	0	0
E11	1	0	0	4	0	3	0	0	0	0	0	0	8
E12	1	0	0	0	0	0	0	0	0	0	0	0	0
E13	0	0	2	0	0	0	0	0	0	0	0	0	0
E14	0	0	3	0	0	0	0	2	0	0	3	0	0
E15	4	0	0	0	0	0	0	0	0	1	0	0	0
E16	0	0	0	1	0	[illegible]	0	0	0	0	0	3	0
E17	1	0	0	1	0	1	1	0	0	0	0	0	7
E18	0	0	0	0	0	1	0	0	1	0	0	0	0
E19	0	1	0	0	0	0	0	1	0	0	1	0	0

O Matrix

E1	6
E2	30
E3	5
E4	3
E5	4
E6	2
E7	5
E8	16
E9	4
E10	4
E11	22
E12	17
E13	3
E14	9
E15	7
E16	5
E17	13
E18	5
E19	23

Figure 3. Example of step1

2) *Certainly existing attacks' class*

Eliminate attacks, which have a possibility of existence equal to 100 %. These attacks haven't a common event with other attacks. In this case, the sum of their occurrence number is less than or equal to the audited occurrences' number for this event. For eliminate these attacks we compare the $AE_{(N_{ap},N_{ep})}$ and matrix $O_{(N_{ep})}$. So, attacks i that verify the following formula:

$$\forall j \in N_e \left(\left(AE_{ij} > 0\right) \rightarrow \left(\left(\sum_{i=0}^{i=N_a} AE_{ij}\right) \le O_j \right) \right) \qquad (14)$$

The result is the attacks noted $\mathrm{N_{ha}}$. Consequently, we resize Matrix attack-event "AE" to the size$\left(\mathrm{N_{ha}}, \mathrm{N_{ep}}\right)$. After these treatments, we eliminate the events j that verifies the formula:

$$\left(\sum_{i=0}^{i=N_a} AE_{ij}\right) \le O_j \qquad (15)$$

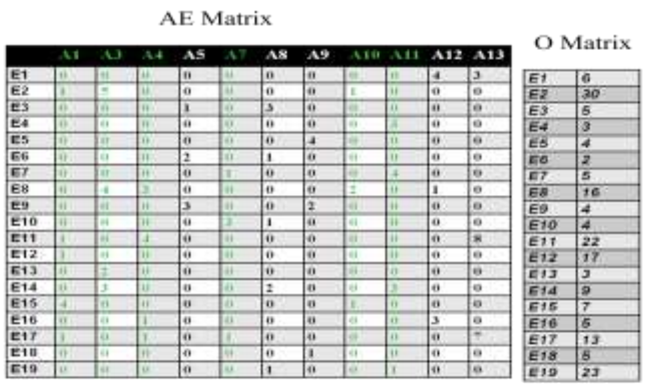
The number of attacks events retain is noted $\mathrm{N_{he}}$.

The results are matrixes with the following dimensions:

$$AE_{(N_{ha},N_{he})}, O_{(N_{he})}, H_{(N_{ha})}, R_{(N_{ha})}$$

(16)

Figure 4 shows an example of elimination certainly existing attacks. For example attack A11 haven't a common event with other attacks for event E4,and when they have a common event with other attacks(E7,E14,E19), the sum of their occurrences number is less than or equal to the audited occurrences number for this event.


AE Matrix

	A1	A3	A4	A5	A7	A8	A9	A10	A11	A12	A13
E1	0	0	0	0	0	0	0	0	0	4	3
E2	1	5	0	0	0	0	0	1	0	0	0
E3	0	0	0	1	0	3	0	0	0	0	0
E4	0	0	0	0	0	0	0	0	3	0	0
E5	0	0	0	0	0	0	4	0	0	0	0
E6	0	0	0	2	0	1	0	0	0	0	0
E7	0	0	0	0	1	0	0	0	4	0	0
E8	0	4	3	0	0	0	0	2	0	1	0
E9	0	0	0	3	0	0	2	0	0	0	0
E10	0	0	0	0	3	1	0	0	0	0	0
E11	1	0	4	0	0	0	0	0	0	0	8
E12	1	0	0	0	0	0	0	0	0	0	0
E13	0	2	0	0	0	0	0	0	0	0	0
E14	0	3	0	0	0	2	0	0	3	0	0
E15	4	0	0	0	0	0	0	1	0	0	0
E16	0	0	1	0	0	0	0	0	0	3	0
E17	1	0	1	0	1	0	0	0	0	0	7
E18	0	0	0	0	0	0	1	0	0	0	0
E19	0	0	0	0	0	1	0	0	1	0	0

O Matrix

E1	6
E2	30
E3	5
E4	3
E5	4
E6	2
E7	5
E8	16
E9	4
E10	4
E11	22
E12	17
E13	3
E14	9
E15	7
E16	5
E17	13
E18	5
E19	23

Figure 4. Example of step 2

3) *Uncertainly existing attacks' class*

This last class is concerned by our contribution. These attacks that we doubt for their existence represent the real Security Audit Trail Analysis Problem (SATAP). These attacks represent the uncertainly existing attacks' class.

B. *Divisions SATAP to sub-SATAP*

We use the 3rd class. So, we regroup the attacks that generate the same kind event where the sum of the occurrences number exceeds the occurrences number audited for this event. This relation between attacks called "mutually exclusive".

Each attack group contain attacks "mutually exclusive" over there and we associate to each attack group the event group which they have an occurrences number higher than the audited occurrences number. We create the Sub-SATAP where each $\mathrm{SATAP_i}$ contains the attacks of the group i with associated events.

Figure.5 presents the associated algorithm to the process described above. Procedure add-element is called in procedure grouping.

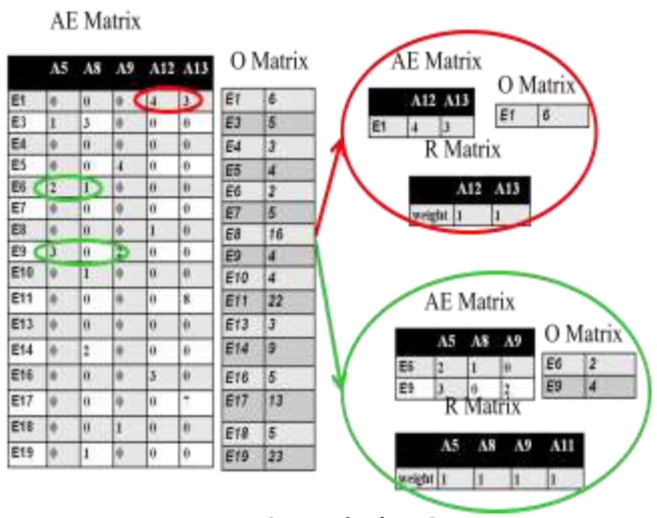
```
Procedure grouping
begin
for i=1 to $N_{ha}$  do
     if (not-marked attack(i)) then
                                         Create group() ;
                                          Mark(i) ;
                                          add-element(i) ;
```

```
end.
Procedure add-element(i)
begin
for $j = 1$ to $N_{ep}$ do
   if $AE_{ij} \neq 0$ then
            for $x = i + 1$ to $N_{ha}$ do
               if $AE_{xj} \neq 0$ then
                  if (not-marked attack (x)) then
                        add-to-group(x);
                        Mark(i);
                        add-element(x);
                  else fusion-group-where-belong(x,i);
end .
```

Figure 5. Division SATAP to Sub-SATAP algorithm

Each sub-SATAP $\{AE, R, O, H\}_i$ defines the sub-SATAP$_i$

Figure 6 shows an example of Divisions SATAP to sub-SATAPs. For example, attack A5 is mutually exclusive with attacks A8 for the event E6 and mutually exclusive with attacks A9 for the event E9. For this reason, the attacks A5 A8 A9 with the event E6 E9 represent one of the sub-SATAP.


AE Matrix

	A5	A8	A9	A12	A13
E1	0	0	0	4	3
E3	1	3	0	0	0
E4	0	0	0	0	0
E5	0	0	4	0	0
E6	2	1	0	0	0
E7	0	0	0	0	0
E8	0	0	0	1	0
E9	3	0	2	0	0
E10	0	1	0	0	0
E11	0	0	0	0	8
E13	0	0	0	0	0
E14	0	2	0	0	0
E16	0	0	0	3	0
E17	0	0	0	0	[illegible]
E18	0	0	1	0	0
E19	0	1	0	0	0

O Matrix

E1	6
E3	5
E4	3
E5	4
E6	2
E7	5
E8	16
E9	4
E10	4
E11	22
E13	3
E14	9
E16	5
E17	13
E18	5
E19	23

AE Matrix

	A12	A13
E1	4	3

O Matrix

E1	6

R Matrix

	A12	A13
weight	1	1

AE Matrix

	A5	A8	A9
E6	2	1	0
E9	3	0	2

O Matrix

E6	2
E9	4

R Matrix

	A5	A8	A9	A11
weight	1	1	1	1

Figure 6. Example of step 3

C. *The crossover*

The proposed crossover operator is a crossover strongly random. All heritage possibilities are reached from the first generation in reduced time. The advantage of this crossover is the minimization of the generation number needed to generate certain individual that can be the best solution of our problem. This crossover consists, firstly, to make a cloning one of the two parents. So, the generated member inherits randomly the genes of the second parent, and we put it in the corresponding locus in the cloned parent [13] as shown in Figure 7.

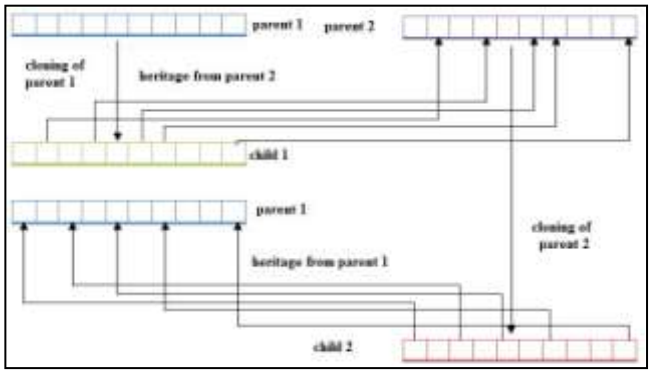


Figure 7. Crossover of our proposition

D. *Resolve the sub SATAP simultaneously with RMI*

This step consists to resolve the sub-SATAPs simultaneously using the remote method invocation. We associate to each sub-SATAP a thread to resolve it in the suitable computer (best performance for the biggest sub-SATAP) as shown in Figure 8.

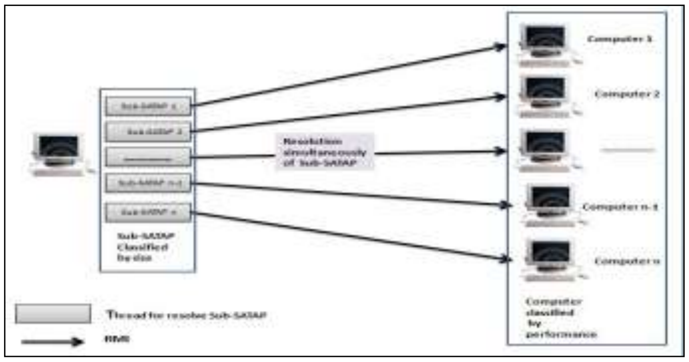

Computer 1
Computer 2
Computer n-1
Computer n
Resolution simultaneously of sub-SATAP
Sub-SATAP Classified by size
Computer classified by performance
Thread for resolve Sub-SATAP
RMI

Figure 8.Simultaneous resolution mechanism of the sub-SATAP

V. EXPERIMENTATIONS

A. *Used Metrics*

To evaluate the performance of this contribution, several tests with several benchmarks extracted from the KDD Cup 1999 data set [14] was performed. The evaluation metrics used are the following:

- False positive: false alarms caused by legitimate changes in program behavior [15].
- False negative: missed intrusions caused by attackers who mimic benign users [15].
- Detection rate.
- Processing time.

TABLE I. IT SUMMARIZES THE DIFFERENT RESULTS OBTAINED WITH THE VARIOUS BENCHMARKS.

Benchmark	False + %	False - %	Detection rate %	Processing Time	Processing Time with RMI
benchmark(2,4	0%	0%	100%	≈0 ms	≈0 ms
benchmark(5,9	0%	0%	100%	≈0 ms	≈0 ms
benchmark(6,9	0%	0%	100%	≈7 ms	≈7 ms
benchmark(9,11)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(10,12)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(15,19)	0%	0%	100%	≈13 ms	≈16 ms
benchmark(15,20)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(17,35)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(21,40)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(24,50)	0%	0%	100%	≈0 ms	≈0 ms
benchmark(25,51)	0%	0%	100%	≈232 ms	≈265 ms

We remark that for all benchmarks the proportion of false positive and false negative equal to 0 % and the detection rate equal to 100 % that signify, the good quality of resultants.

We remark also that there are several benchmarks, treated in real time (0 ms). This means, that during the two first steps of attack classification we can attest about the existing attacks or not. The other benchmarks are concerned by the second step “Divisions SATAP to sub-SATAP” and the genetic algorithms must be applied to identify attacks that justify the increase of processing time. The benchmark (15, 19) and (25, 51) are treated in the more reducing time than the resolution without remote method invocation due to simultaneously treatment of the sub-SATAP.

VI. COMPARATIVE STUDY

First we compare the results of the contribution and the work of Mé [7] using the same benchmarks. The following metric are used: the number of detected attacks, the number of constraints raped during each generation, the convergence speed to the best solution, the number of generation and the necessary time for the resolution.

Results show in Figure.10 and Figure.9 and table 2 that with our work we detect the same attack's percentage that represents the real attacks.

However, there are some differences:

- Runtime: the processing time of our proposition is less than the processing time of [7] and [11] due to minimizing of problem size and dividing the problem to sub-problems and the simultaneously treatment of sub-SATAP.
- Generations number: the number of generations needed for our proposition is less than the number of generations needed for [7] and [11], because the size of the biggest sub-SATAP to be treated is less than or equal (in the worst case) the size of SATAP.
- Convergence speed: the convergence speed of our proposition is faster than [7]and[11], because the resolution of sub-SATAP is more efficient than that of SATAP.(in the worst case) the size of SATAP.
- Constraints' violation: due to the filtering operation and the dividing of SATAP to sub-SATAP, the constraint's violation of contribution is lesser (almost nonexistent) than the [7] and [11].

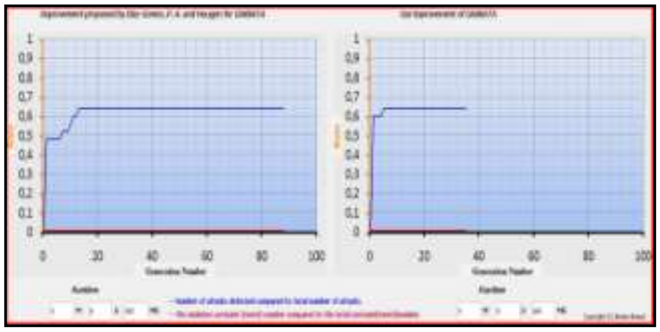


Figure 9. Comparison between our improvement and Diaz-Gomez, P. A. and Hougen for benchmark (15,19)

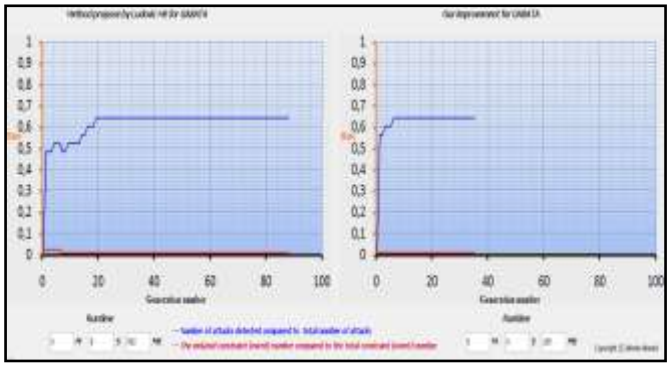


Figure 10. Comparison between improvement and L.Mé resolution for benchmark (25,51)

TABLE II. TCOMPARISON BETWEEN THE TWO RESOLUTION METHODS OF SATAP

	Classical resolution	Our proposition	
		Without RMI	With RMI
benchmark(2,4)	≈16 ms	≈0 ms	≈0 ms
benchmark(5,9)	≈62 ms	≈0 ms	≈0 ms
benchmark(6,9)	≈125 ms	≈7 ms	≈7 ms
benchmark(9,11)	≈234 ms	≈0 ms	≈0 ms
benchmark(10,12)	≈312 ms	≈0 ms	≈0 ms
benchmark(15,19)	≈1s 60 ms	≈16 ms	≈13 ms
benchmark(15,20)	≈1s 139 ms	≈0 ms	≈0 ms
benchmark(17,35)	≈2s 777 ms	≈0 ms	≈0 ms
benchmark(21,40)	≈4s 771 ms	≈0 ms	≈0 ms
benchmark(24,50)	≈7s 909 ms	≈0 ms	≈0 ms
benchmark(25,51)	≈8s 502 ms	≈265 ms	≈232 ms

VII. CONCLUSIONS AND PERSPECTIVES

Using Genetic Algorithms for the Security Audit Trail Analysis Problem has significant results. This contribution consists to classify attacks in Security Audit Trail in three classes and divide the 3rd class to sub-problems. Then, we apply the genetic algorithm with the proposed crossover operator in [13] and same selective function of Mé [7]. The second contribution is to optimize the resolution time of genetic algorithm. For this we apply RMI (remote method invocation) to each sub-problem simultaneously.

The contribution brings the following advantages:

- 0% False +.
- 0% False -.
- 100% detection rate.
- Minimizing the runtime.
- Increasing the convergence speed.
- Reducing the constraints violation.
- Reducing the generations number needed to solve this problem.

This improvement confirms the power of using Genetic Algorithms for the Security Audit Trail Analysis Problem where we detect 100% of real attacks.

Our perspective is to propose architecture of multi-agents system for real time resolution of SATAP.

REFERENCES

[1] E. Cole, Ronald L. Krutz, J. Conley, “Network Security Bible,” Wiley Publishing, Inc.January 2005 ISBN13:978-0-7645-7397-2

[2] L. Mé ,“Un algorithme génétique pour détecter des intrusions dans un système informatique ,” *VALGO*, 95(1):68-78, 1995.

[3] R. Bace, P. Mell , “NIST Special Publication on Intrusion Detection Systems” ,2001 .

[4] J. Anderson, , ”Computer Security Threat Monitoring and Surveillance”. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania (1980).

[5] D. Denning,”An Intrusion-Détection Model”. IEEE transaction on Software Engineering, 13(2):222-232 (1987).

[6] H. Debar, M. Dacier, A. Wespi, “A Revised Taxonomy for Intrusion-Detection Systems”. Annales des Télécommunications, 55(7-8), 2000.

[7] L. Mé, “GASSATA, A Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis”. Web proceedings of the First international workshop on the Récent Advances in Intrusion Détection 1998 .

[8] P. A. Diaz-Gomez, D. F. Hougen, “Improved Off-Line Intrusion Detection using a Genetic Algorithm” In Proceedings of the Seventh International Conference on Enterprise Information Systems, 2005

[9] P. A. Diaz-Gomez, D. F. Hougen,, “Analysis and Mathematical Justification of a Fitness Function used in an Intrusion Detection System” In Proceedings of the Seventh Annual Genetic and Evolutionary Computation Conference 2005.

[10] P. A. Diaz-Gomez, D. F. Hougen, “Mathematical Justification of a Fitness Function used for Audit Trail Intrusion Analysis” In the Poster Session of the Research Experience Symposium 2005, at the University of Oklahoma.

[11] P. A. Diaz-Gomez, D. F. Hougen, “A Case Study in Genetic Algorithms applied to Off-line Intrusion Detection Systems” In 1st Annual Computer Science Research Conference at the Stephenson Research and Technology Center 2005.

[12] P. A. Diaz-Gomez, D. F. Hougen, “Further Analysis of an Off-Line Intrusion Detection System”: An Expanded Case Study in Multi-Objective Genetic Algorithms SCISS'05 The South Central Information Security Symposium

[13] M. Rachid,N. Ghoualmi, “Crossover and Mutation Based Cloning Parent for Degree Constrained Minimum Spanning Tree Problem,” , AMERICAN UNIVERSITY OF SHARGAH, UAE , IEEE ICESMA (2010), 30-1 April, ISBN: 978-9948-427-14-8.

[14] KDD Cup Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html October 28, 1999.

[15] P. Roberto, M. Luigi V, “Intrusion Detection Systems,” 2008, ISBN:978-0-387-77265-3.