A New Off-Line Intrusion Detection System

M. Daoudi, A. Boukra
Faculty of Electronics and Computer Science
Laboratory LSI, USTHB
BP 32 16111 El Alia, Bab-Ezzouar
Algiers, Algeria
{dmfinfo, amboukra}@yahoo.fr

ABSTRACT: *The problem of intrusion detection is studied extensively in computer security. The development of security mechanisms, like Intrusion Detection Systems, is of great importance in order to preserve the confidentiality, integrity and availability of data stored in computers. Off-line intrusion detection can be accomplished by searching audit trail logs of user activities for matches to patterns of events required for known attacks. This is a combinatorial optimization problem, the NP-complete. Metaheuristics offer an alternative to solve this type of problems as databases of events and attacks grow. This paper presents an approach using an evolutionary algorithm, Harmony Search metaheuristic, to detect intrusions. Experiments are performed to show its effectiveness to detect the unseen intrusion attacks with high detection rate and recognize normal network traffic with low false alarm rate. Furthermore, comparisons with a "Biogeography" inspired intrusion detection approach are made. The results confirm the good behavior of our new approach.*

Keywords: Metaheuristics, Harmony Search, Biogeography Based Optimization, Evolutionary Algorithm, Intrusion Detection, Security Audit

Received: 12 March 2013, Revised 10 May 2013, Accepted 15 May 2013

1. Introduction

With the rapid expansion of computer networks during the past few years, security has become a crucial issue for modern computer systems. Different security mechanisms have been developed in order to preserve the confidentiality, integrity and availability of data stored in computers, using different techniques and methods, like authentication, cryptography, firewalls, proxies, antivirus, Virtual Private Network (VPN), Intrusion Detection System (IDS) [1], [2]. This paper focuses on Intrusion Detection Systems using the audit trail file.

An intrusion detection system dynamically monitors the events taking place in a system, and decides whether these events are symptomatic of an attack or constitute a legitimate use of the system [3], on the hypothesis that exploitation of a system's vulnerabilities is based in the abnormal use of the system [4]. The detection method is one of the principal characters of classification of IDSs. When the IDS uses information about the attacks (misuse detection or scenario approach), we qualify it as knowledge-based. When the IDS uses information about the normal behavior of the system it monitors (anomaly detection or behavioral approach), we qualify it as behavior-based. However, both approaches may be complementary [5].

Misuse detection identifies intrusions by matching observed data with pre-defined descriptions of intrusive behavior. Therefore,

well-known intrusions can be detected efficiently with a very low false alarm rate. However, Misuse detection will fail easily when facing unknown intrusions. One way to address this problem is to regularly update the knowledge base, either manually which is time consuming and laborious, or automatically with the help of supervised learning algorithms. Unfortunately, datasets for this purpose are usually expensive to prepare, as they require labeling of each instance in the dataset as normal or a type of intrusion.

The behavioral approach was introduced by Anderson [6], and extended by Denning [4]. It hypothesizes that abnormal behavior is rare and different from normal behavior. Hence, it builds models for normal behavior and detects anomaly in observed data by noticing deviations from these models. The techniques developed in behavioral approach include the expert systems, statistical models [7], and artificial immune systems [8], [9]. Other techniques like Bayesian parameter estimation [10] and clustering [11]-[14], Genetic Algorithms [15], [16] are also used. Clearly, anomaly detection has the capability of detecting new types of intrusions, and only requires normal data when building profiles. However, its major difficulty lies in discovering boundaries between normal and abnormal behavior, due to the deficiency of abnormal samples in the training phase. Another difficulty is to adapt to constantly changing normal behavior, especially for dynamic anomaly detection.

In addition to the detection method, there are other characteristics one can use to classify IDSs. They can be classified as host-based, multihostbased and network-based [2]. Hostbased IDSs monitor a single computer using the audit trail of the operating system whereas network-based IDSs monitor computers on a network by scrutinizing the audit trail of multiple hosts and network traffic. A multihost-based IDS analyzes data from multiple computers.

This paper deals with a host-based IDS by security audit trail analysis. It can be performed by searching audit trail logs of user activities for predefined attacks. This is a combinatorial optimization problem NP-Complete [17], heuristic methods will need to be used as databases of events and attacks grow. We consider Harmony Search metaheuristic (HS) [18] as detection engine. HS is a population-based evolutionary algorithm taking inspiration from the music improvisation process, where musicians improvise their instruments' pitches searching for a perfect state of harmony. It has proven its abilities in solving various optimization problems [19], [20]. The effectiveness of our approach is evaluated by its ability to make correct predictions. Comparisons with a biogeography inspired intrusion detection approach [21], [22] are performed. We organized the rest of the paper as follows: The second section presents a formalization of the Security Audit Trail Analysis Problem as well as some related work. Our contribution using HS for misuse detection is presented in section 3. Comparisons with a biogeography inspired intrusion detection approach is made in section 4. Finally, the conclusion presents the advantages of our approach, and the prospects work.

2. Security Audit Trail Analysis Problem and Related Work

Some researchers are trying to analyze the problem of intrusion detection by using a multiple fault diagnosis approach, somewhat analogous to the process of a human being diagnosed by a physician when suffering from a disease. In this approach, the attack scenarios are modeled as a set of couples (E_i ,N_i) where E_i the type of event and N_i is the number of occurrences of this type of event in the scenario. An events-attacks matrix is defined, which is known as pre-learned domain knowledge (analogous to knowledge possessed by a physician). The occurrence of one or more attacks is required to be inferred from newly observed events (analogous to symptoms). Such a problem is reducible to a zero-one integer problem, which is NP-Complete [17].

The mathematical model is as follows [23]:

$$\begin{cases} Max \sum_{j}^{N_a} R_j * H_j \\ (AE * H)_i \le O_i, i \in \{1...N_e\} \end{cases} \quad (1)$$

Where:

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

i generated by the attack *j* ($AE_{ij} \geq 0$)

• *R* the N_a- dimensional weight vector, where R_i ($R_i > 0$) is the weight associated with the attack *i* (R_i is proportional to the inherent risk in attack scenario *i*)

• *O* the N_e - dimensional vector where O_i is the number of events of type *i* present in the audit trail (*O* is the observed audit vector)

• *H* the N_a- dimensional hypothesis vector, where $H_j = 1$ if attack *i* is present according to the hypothesis and $H_j = 0$ otherwise (*H* describes a particular attack subset).

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 product *R* * *H* (it is the pessimistic approach: finding *H* so that the risk is the greatest), subject to the constraint $(AE * H)_i \leq O_i, 1 \leq i \leq N_e$. Because finding *H* vector is *NP*-Complete, the application of classical algorithm is impossible where Na equals to several hundreds. The heuristic approach 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.

To derive a new hypothesis based on the past hypothesis, Biogeography Based Optimization algorithm (BBO) and Genetic Algorithms (GA) are employed as optimization component in previous works [21]-[26]. All works coded solutions in binary strings, where the length of a string was the number of attacks, and 1's or 0's in a genome indicated if an attack was present. An overview of Harmony Search metaheuristic is presented in the first part of the next section. Its adaptation in our offline intrusion detection system is given in part two.

3. Harmony Search-Based Security Audit Trail Analysis Approach

3.1 Harmony Search Metaheuristic overview

Harmony Search was devised as a new metaheuristic algorithm, taking inspiration from the music improvisation process, where musicians improvise their instruments' pitches searching for a perfect state of harmony. Analogies with optimization process are such that:

- The instrument *i* corresponds to the decision variable x_i for $i = 1, 2, \ldots, n$

- A music note coming out of the instrument *i* is analogous to the value of variable x_i

- A Harmony corresponds to a solution vector

- Musical aesthetic is analogous to the Fitness

The standard HS algorithm consists of the following steps [27]:

1) Initialization of the optimization problem and algorithm parameters:
The optimization problem is specified as follows:

$$\textit{Minimize (or Maximize)}\ f(x)$$
$$\text{subjected to } x_i \in X_i,\ i = 1, 2, \ldots, n.$$

where $f(.)$ is a scalar objective function to be optimized; *x* is a solution vector composed of decision variables x_i, $i = 1, 2, .., n$; X_i is the se:t of possible range of values for each decision variable x_i, that is, ${}_L x_i \leq X_i \leq {}_U x_i$, where ${}_L x_i$ and ${}_U x_i$ are the lower and upper bounds for each decision variable in the case of continuous decision variables and $x_i \in \{x_i(1), \ldots, x_i(k), \ldots, x_i(K)\}$ when the decision variables are discrete. *N* is the number of decision variables. In addition, the control parameters of HS are also specified in this step. They are the Harmony Memory Size (HMS) i.e., the number of solution vectors (population members) in the HM(in each generation); the HM considering rate (HMCR); the pitch-adjusting rate (PAR) and the number of improvisations (NI) or stopping criterion.

3.2 Harmony Memory (HM) initialization

Each component of each solution vector in the parental population (HM) is randomly chosen. Then, obtained solutions are

reordered in terms of the objective function value: $f(x^1) \leq f(x^2) \ldots \leq f(x^{HMS})$, where $x^i = (x_1^i, x_2^i, \ldots, x_n^i)$ is the i^{th} solution vector. Harmony is represented by the matrix:

$$HM = \begin{bmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^{HMS} & \cdots & x_n^{HMS} \end{bmatrix} \quad (2)$$

3.3 New harmony improvisation

A new harmony vector $x' = (x_1', \ldots, x_n')$ is generated based on three rules: memory consideration; pitch adjustment; and random selection. In the memory consideration, the value of the ith decision variable in the new vector x' is chosen from any of the existing values in the i^{th} column of the current HM matrix with probability HMCR. Note that (1 - HMCR) is the rate of randomly selecting a fresh value from the possible range of values in X_i. Then, every component obtained after the memory consideration procedure is further examined to determine whether it should be pitch adjusted. This operation uses the parameter PAR (which is the rate of pitch adjustment). If the new harmony vector x' is better than the worst harmony in the HM, judged in terms of the objective function value, the new harmony is included in the HM, and the existing worst harmony is excluded from the HM.

3.4 Check stopping criterion

If the stopping criterion (maximum NI) is satisfied, the computation is terminated. Otherwise, new harmony improvisation step is repeated.

However, the HS standard method, as given in [27] is not entirely appropriate in the treatment of discrete binary problems as it is the case with our particular problem. An extension of HS to the effective resolution of binary problems, proposed in [28], is used.

3.2 Security Audit Trail Analysis with Harmony Search

The approach aims to determine if the events generated by a user correspond to known attacks, and to search in the audit trail file for the occurrence of attacks using HS algorithm. The goal of the heuristic is to find the hypothesized vector H that maximizes the product $R * H$, subject to the constraint $(AE * H)_i \leq O_i, 1 \leq i \leq N_e$, where R is a weight vector that reflects the priorities of the security manager, AE is the attack-events matrix that correlates sets of events with known attacks and N_e the number of types of audit events.

HS method is based on a stochastic optimization process and on the musical performance process of finding the perfect harmony in a musical orchestra where each musician plays a note to find a better harmony. We are in a situation where the coding of harmonies is immediate since the solution of the problem is expressed specifically in the form of a binary sequence. A harmony is considered as a series of N_a notes with values 0 or 1. In other words, the note i of the harmony will be 1 if the harmony is a solution in which the attack i is declared present. Otherwise, this note takes the value 0. Each harmony is a particular instance of the vector H (the sum of its components corresponds to the number of detected attacks).

As we have to solve a maximization problem, the best solution (harmony) is associated with the hypothesis $H = (H_1, \ldots, H_{Na})$ of larger value of the function:

$$F(H) = \sum_1^{Na} Ri * Hi \quad (3)$$

which represents the total risks incurred by the system under surveillance. In addition, as we deal with a constrained problem, any harmony, solution to the problem, must satisfy the constraints $(AE \times H)_i \leq O_i$, $0 < i < N_e$.

This fitness function does not however, take into account the constraint feature of our problem. As a large number of harmonies do not respect the constraint, we decided to penalize them by reducing their fitness values [27]. A penalty function (P) is computed. It increases as the esthetic of the harmony decreases (given the content of the observed audit matrix O). Let Te represents the number of types of events for which $(AE \times H)_i > O_i$, then the corresponding penalty function is given by:

$$F(H) = \alpha + (\sum_1^{Na} Ri * Hi - \beta * T_e^2)$$

Where α and β are two parameters such that β makes it possible to modify the slope of the penalty function and α sets a

threshold making the fitness positive. If α negative fitness value is found, it is equaled to 0 and the corresponding individual cannot be selected. So the α parameter allows the elimination of too unrealistic hypothesis.

Now that the harmony and the objective function are well defined, the rules used to construct the new solution (new harmony improvisation) during the optimization process evolution, that are memory consideration and pitch adjustment mentioned in section 3.1, are [28]:

• **Memory consideration rule:**

For $k = 1, \ldots, N_a$, the binary variable x_k is such that :

$$x_k = \begin{cases} HM_{tk}, t \in \{1, ..., HMS\} \text{ if } r_1 < HMCR \\ R \qquad\qquad otherwise \end{cases} \tag{5}$$

where :

$$R = \begin{cases} 0 & if\ r_2 < 0.5 \\ 1 & otherwise \end{cases}$$

and r_1, r_2 are two independant random numbers between 0 and 1; t is an integer randomly chosen in the set: {1,……, HMS}.

• **Pitch adjustment rule**

The adjustment operator is used to find better solutions locally. Given a new solution $x = (x_1, \ldots, x_k, \ldots, x_{Na})$, the value of each variable is modified with a probability PAR such that:

$$x_k = \begin{cases} HM_{bk} & \text{if } r < PAR \\ x_k & otherwise \end{cases} \tag{6}$$

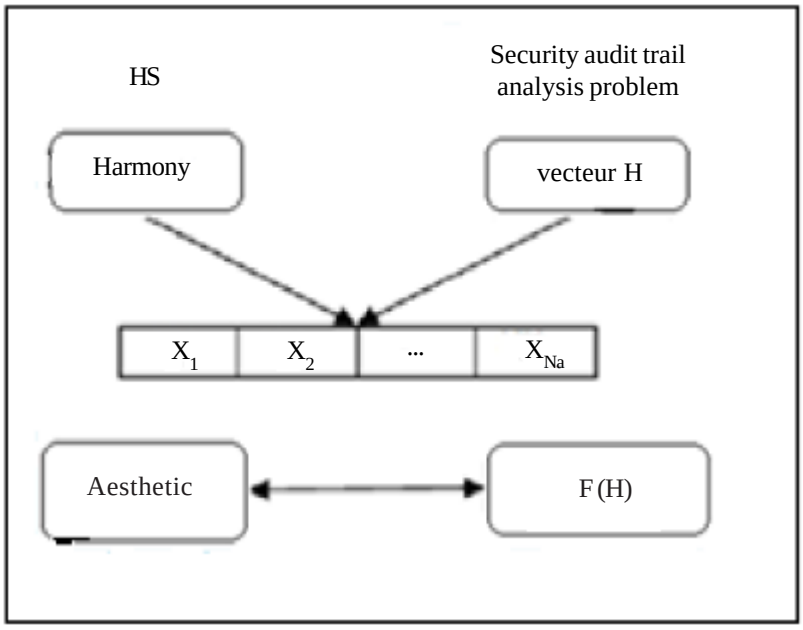

Figure 1. Coding of solutions and the objective function

where: r is a random number between 0 and 1, HM_{bk} represents the value of the corresponding element in the vector of the global optimum in HM.

Thus, if the new harmony vector x = (x_1, ...,x_{Na}) is better than the worst harmony vector in HM, in terms of the objective function value, it is included in HM and the worst harmony in HM is excluded.

It is now necessary to correctly represent the matrix HM. It is a (*HMS* * *Na*) - matrix, where Na is the number of predefined attacks and HMS the population size parameter. We associate to each row i of the matrix HM (a solution vector to our problem) a value of the objective function *F*. Figure 1 gives a representation of coding of solutions and the objective function. The following pseudo-code shows the different steps when analyzing the security audit trail with HS algorithm. The different parameters HMS, HMCR, PAR and NI must be initialized. Then, HMS initial solution vectors are randomly generated and saved in the HM matrix. The fitness value $F(i)$, corresponding to the ith solution i = 1, .., HMS, is evaluated. Then, the optimization process evolves. It generates a new solution $x = (x_1, ..., x_n)$ from the HM matrix, using the parameters HMCR and PAR. These parameters help the algorithm to obtain better solutions locally or globally.

Pseudo-code of the security audit trail analysis process with HS

Input : ($N_e \times N_a$) - AE matrix and N_e - *O* vector
Initialize parameters NI, HMS, PAR and HMCR
Generate randomly a harmony memory HM (with HMS initial solutions)
Evaluate the fitness of the generated harmonies
IterationNumber = 0
If IterationNumber < *NI* **Then**
  Build new harmony $\boldsymbol{x} = (x1, \ldots, x_k, \ldots, x_{Na})$
  Check constraints
  Compute fitness of $\boldsymbol{x}$
  Get worst harmony in Harmony Memory
  Update harmony memory
  Save best harmony (*H* vector)
  IterationNumber = IterationNumber + 1
EndIf
Output: *H* vector maximizing product *R* * *H* and verifying the constraints.

4. Experimental results

In this research we used the Sun audit trail file [17]. Four (4) types of users were identified: inexperienced novice developer, professional users, and heavy users of UNIX. Some examples of attack-events considered include:

• Trying to change sensitive information on records of files requiring higher privilege,

• Killing critical processes,

• Trying to access different user's files,

• Probing the system,

• Installing of unauthorized, potentially damaging software, and

• Exploiting a security vulnerability to gain higher or different privileges.

We considered first the 24 × 28 - matrix AE (Attacks-Events) [23] with 24 attack scenarios and 28 types of auditable events specific to a Unix system. During the simulations, all the attacks actually present in the analyzed audit file must be known in advance. Thus, the events corresponding to one or more attacks are included in the observed audit vector O. Each of the experiments conducted is characterized by the 5-tuple (*NI*, *HMS*, *HMCR*, *PAR*, *Ia*), where *NI* is the number of generations, *HMS* the population size, HMCR the harmony memory consideration rate, PAR the adjustment rate and Ia the number of attacks actually present in the audit file (number of attacks injected).

Ratios TPR, FPR, Accuracy and Precision [21] are used to evaluate our approach intrusion detection quality. Many tests are performed using Attack-Events matrices of different sizes. All results are obtained as the average of 10 executions carried out for the same better value of the 5-tuple (*NI*, *HMS*, *HMCR*, *PAR*, I_a).

We observe that after a certain number of generations, all injected attacks are detected, and no false attack (TPR = 1 and FPR = 0). In addition, the number of attacks injected has no influence on these results. Best results are obtained with HMS value between 20 and 30. Further, the study of the influence of the two rates HMCR and PAR is important because they contribute significantly, in the algorithm, in finding the best solution. There is a strong correlation between HMCR and PAR on the Harmony Search metaheuristic optimization process. To study their influence on the quality of intrusion detection, we varied the parameter PAR between 0.1 and 0.9 with a step of 0.2; HMCR is selected it in the set {0, 0.3, 0.5, 0.7 , 0.9, 0.98}. We observe that the best results are obtained for values of HMCR and PAR, such as: $0.8 \leq HMCR \leq 0.98$ and $PAR \geq 0.3$.

Table 1 shows the quality of our intrusion detection approach performed on data (attack-events matrix of size 28×24) issued from [23] with parameter values: HMS = 30, HMCR = 0.98, PAR = 0.3 and $I_a = 24$. The chosen parameters for the penalty function are: $\alpha = 300$ and $\beta = 1$. We observe that all the injected attacks are detected after 0.064 seconds (execution time).

Generation number (NI)	Execution time (sec)	Detected attacks number	TPR %	FPR %	Precision %
1	0.002	18	66.67	0	100
5	0.003	18	70.83	0	100
50	0.043	20	83.83	0	100
100	0.060	22	91.67	0	100
200	**0.064**	**24**	**100**	**0**	**100**
220	0.066	24	100	0	100
250	0.077	24	100	0	100

Table 1. Intrusion detection quality (28×24 – AE matrix)

The results given in Table 2 concern tests performed on data with an attack-events matrix of dimension (100×200). The different parameters are such that: HMS = 30, HMCR = 0.98, PAR = 0.3. The number of injected attacks is: $I_a = 200$. The chosen parameters for the penalty function are: $\alpha = 470$ and $\alpha = 13$. All attacks are detected after 0.763 seconds (execution time).

Generation number (NI)	Execution time (sec)	Detected attacks number	TPR %	FPR %	Precision %
0	0.024	118	59	0	100
100	0.082	128	64	0	100
200	0.138	142	71	0	100
500	0.326	168	84	0	100
700	0.400	177	88.5	0	100
1000	0.478	197	98.5	0	100
1700	**0.763**	**200**	**100**	**0**	**100**

Table 2. Intrusion detection quality (100×200 – AE matrix)

5. Comparisons with a biogeography inspired intrusion detection approach

Biogeography-based optimization (BBO) was first presented by D. Simon in 2008 [29]. It is an evolutionary algorithm based on the science of biogeography [31], which is the study of the geographical distribution of organisms.

In BBO, problem solutions are represented as islands, and the sharing of features between solutions is represented as migration between islands. Islands that are well suited as habitats for biological species are said to have a high island suitability index (ISI). Features that correlate with ISI include rainfall, topographic diversity, area, temperature, etc. The variables that characterize these features are called suitability index variables (SIVs). SIVs are the independent variables of the island, and ISI is the dependent variable. Islands with a high ISI tend to have a large number of species, and those with a low ISI have a small number of species. Islands with a high ISI have many species that emigrate to nearby islands because of the accumulation of random effects on its large populations. Emigration occurs as animals ride flotsam, fly, or swim to neighboring islands. Suppose that we have some problem, and that we also have several candidate solutions. A good solution is analogous to an island with a high ISI, and a poor solution is like an island with a low ISI. High ISI solutions are more likely to share their features with other solutions, and low ISI solutions are more likely to accept shared features from other solutions.

An intrusion detection approach with BBO has been developed [22]. We compare the latter with our new method. Tests are performed on different data. Results reported in Table 3 concern a (28×24) - Attack-Events matrix issued from [23] with 24 attacks injected, a AE matrix of size (100×300) with 300 attacks injected and finally an AE matrix of size (100×700) with 700 attacks injected. Note that the tests with BBO and HS were performed on the same machine.

The different parameters involved in BBO algorithm are set to: Mutation rate = 0.005, Population size = 50 and Elitism value = 2. In our HS based method, the parameters values are: HMS = 30, HMCR = 0.98 and PAR = 0.3. The chosen parameters for the penalty function are: $\alpha = 470$ and $\beta = 13$.

In both approaches, all attacks are detected. However, we clearly observe that the intrusion detection process duration for all attacks is better using our new approach, and the difference increases as the AE matrix size grows.

Method used	AE (24×28)		AE (100×300)		AE (100×700)	
	Nb of detected attacks	Execution time (sec)	Nb of detected attacks	Execution time (sec)	Nb of detected attacks	Executiontime time (sec)
BBO	24	0.109	300	9.640	700	581.34
HS	24	0.056	300	3.973	700	7.167

Table 3. HS-based approach vs. BBO-based approach

6. Conclusion

Security Audit trail Analysis can be accomplished by searching audit trail logs of user activities for known attacks. The problem is a combinatorial optimization problem NP-Hard. Metaheuristics offer an alternative for solving this type of problem when the size of the database events and attacks grow. We propose to use Harmony Search metaheuristic as detection engine.

Experimental results of simulated intrusions detection are given. The effectiveness of the approach is evaluated by its ability to make correct predictions. It proved to be effective and capable of producing a reliable method for intrusion detection. The results indeed show the good performance of the proposed approach. An important result was the consistency of results, independently of the number of attacks actually present in the analyzed audit file. This means that the performance of the detection system is not deteriorated in the case of multiple attacks. The execution time is very satisfying. All this allows us to consider that the proposed approach constitutes an efficient and reliable intrusion detection system.

Comparisons with a biogeography inspired approach is made. We observe clearly that the intrusion detection duration of all attacks are significantly lower when using our new HS-based approach. However, these systems are usually developed for predefined environments and do not offer a solution to some network characteristics such as changes in behavior of users and services, the increasing complexity and evolution of the types of attacks that they may be subject, the speed of attacks that can occur simultaneously on several machines, etc.

References

[1] Cole, E., Krutz, R. L., Conley, J. (2005). Network Security Bible, Wiley Publishing.

[2] Tjaden, B. C. (2004). Fundamentals of Secure Computer Systems. Franklin and Beedle & Associates.

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

[4] Denning, D. (1987). An Intrusion-Detection Model. *In*: Proc. of the 1986 IEEE Symposium on Security and Privacy, p. 118-131.

[5] Tombini, E. (2006). Amélioration du Diagnostic en Détection d'Intrusions: Etude et Application d'une Combinaison de Méthodes Comportementale et par Scénarios. Thèse de Doctorat de l'Institut National des Sciences Appliquées de Rennes.

[6] Anderson, J. (1980). Computer Security Threat Monitoring and Surveillance. Technical Report 79F296400, James P. Anderson, Co., Fort Washington, PA.

[7] Bace, R. G. (2000). Intrusion Detection Systems. Mac Millan Technique Publication, USA.

[8] Haidong, Yang., Jianhua, Guo., Feiqi, Deng. (2011). Collaborative RFID Intrusion Detection with an Artificial Immune System. *Journal of Intelligent Information System*, 36 (1) 1-26.

[9] Bankovi, Z., Álvaro, A., de Goyeneche, J. M. (2009). Intrusion Detection in Sensor Networks Using Clustering And Immune systems. Lecture Notes in Computer Science, V. 5788, Intelligent Data Engineering and Automated Learning - IDEAL 2009, p. 408-415.

[10] Cho, S. Cha, S., SAD. (2004). Web Session Anomaly Detection Based on Parameter Estimation. *Computers & Security*, 23 (4) 265-351.

[11] Xu, B., Zhang, A. (2005). Application of Support Vector Clustering Algorithm to Network Intrusion Detection. *In*: Proc. of the International Conference on Neural Networks and Brain, ICNN&B '05. 2, p. 1036 – 1040.

[12] SH, O., WS, L. (2003). An anomaly Intrusion Detection Method by Clustering Normal User Behavior. Computers & Security, 22 (7) 596-612.

[13] Leon, E., Nasraoui, O., Gomez, J. (2004). Anomaly Detection Based on Unsupervised Niche Clustering with Application to Network Intrusion Detection. *In*: Proc. of the IEEE Conference on Evolutionary Computation (CEC), p. 502-508.

[14] Guan, Y., Ghorbani, A., Belacel, N. (2003). Y-MEANS: a Clustering Method for Intrusion Detection. *In*: Proc. of the Canadian Conference on Electrical and Computer Engineering, p. 1083-1086.

[15] Saniee Abadeh, M., Habibi, J., Lucas, C. (2007). Intrusion Detection Using a Fuzzy Genetics-Based Learning Algorithm. *Journal of Network and Computer Applications*, p. 414-428.

[16] Ozyer, T., Alhajj, R., Barker, K. (2007). Intrusion Detection by Integrating Boosting Genetic Fuzzy Classifier and Data Mining Criteria for Rule Pre-Screening. *Journal of Network and Computer Applications*, p. 99–113.

[17] Mé, L. (1994). Audit de Sécurité par Algorithmes Génétiques. Thèse de Doctorat de l'Institut de Formation Supérieure en Informatique et Communication DE Rennes.

[18] Geem, Z. W., Kim, J. H., Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search, *Simulations* 76, 60-68.

[19] Coelho, L. S, Andrade Bernert, D. L. (2008). An improved harmony search algorithm for synchronization of discrete-time chaotic systems. Industrial and Systems Engineering Program, LAS/PPGEPS, Pontifical Catholic University of Paraná, PUCPR, Imaculada Conceição, 1155, 80215-901 Curitiba, Paraná, Brazil.

[20] Das, S., Mukhopadhyay, A., Roy A, Abraham, A., Panigrahi, B. K. (2010). Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for Global Numerical Optimization. *IEEE Transactions on systems*, man, and cybernetics—part b: cybernetics. 2010 IEEE.

[21] Daoudi, M., Boukra, A., Ahmed-Nacer, M. (2011). A Biogeography Inspired Approach for Security Audit Trail Analysis. *In*: *Journal of Intelligent Computing*, 2 (4) December.

[22] Daoudi, M., Boukra, A., Ahmed-Nacer, M. (2011) Security Audit Trail Analysis with Biogeography Based Optimization Metaheuristic. *In*: Proceedings of the International Conference on Informatics Engineering & Information Science: ICIES, A. Abd Manaf et al. (Eds.): ICIEIS, Part II, CCIS 252, p. 218–22.© Springer-Verlag Berlin Heidelberg.

[23] Mé, L. (1998). GASSATA, a Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis. *In*: Proc. of the 1st International Workshop on the Recent Advances in Intrusion Detection (RAID 98). Louvain-la-Neuve, Belgium, pages 14–16.

[24] Diaz-Gomez, P. A., Hougen, D. F. (2005). Improved Off-Line Intrusion Detection using a Genetic Algorithm. *In*: Proc. of the seventh International Conference on Enterprise Information Systems, p. 66-73.

[25] Diaz-Gomez, P. A., Hougen, D. F. (2006). A genetic algorithm approach for doing misuse detection in audit trail files. *In*: Proc. of the (CIC-2006), *International Conference on Computing* (CIC-2006), p. 329-335.

[26] Ahmim, A., Ghoualmi, N., Kahya, N. (2011). Improved Off-Line Intrusion Detection using a Genetic Algorithm and RMI. *In*: Proc. of the International Journal of Advanced Computer Science and Applications (IJACSA), 2 (1).

[27] Swagatam, D., Arpan, M., Anwit, R., Ajith, A., Bijaya, K., Panigrahi. (2010). Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for global Numerical Optimization . *IEEE Transactions on Systems, Man, and cybernetics-PartB: cybernetics*.

[28] Ling, Wang., Yin, Xu., Yunfei, M., Minrui, F. (2007). Discrete Harmony Search Algorithm. Shanghai Key. Laboratory of Power Station Automation Technology, School of Mechatronics and Automation, Shanghai University,Shanghai.

[29] Simon, D. (2008). Biogeography-Based Optimization, *IEEE Transactions on Evolutionary Computation*, 12, p. 702-713, December.

[30] Mac Arthur, R., Wilson, E. (1967). The Theory of Biogeography. Princeton, NJ: Princeton Univ. Press.