Learning Problem and BCJR Decoding Algorithm in Anomaly-based Intrusion Detection Systems

Veselina G. Jecheva
Burgas Free University, Burgas, Bulgaria
Email: vessi@bfu.bg

Evgeniya P. Nikolova
Burgas Free University, Burgas, Bulgaria
Email: enikolova@bfu.bg

***Abstract*— The anomaly-based intrusion detection systems examine current system activity do find deviations from normal system activity. The present paper proposes a method for normal activity description using the Hidden Markov Models (HMM), which is tuned up using the gradient based method. The obtained model is utilized as a baseline, depicting the normal system activity. The main purpose is to distinguish the normal traces of user activity from abnormal ones using the BCJR decoding algorithm. Some results from the conducted simulation experiments are introduced as well.**

***Index Terms*— intrusion detection, anomaly-based intrusion detection, learning problem, Hidden Markov Model, BCJR decoding algorithm.**

I. INTRODUCTION

Network computer systems vulnerabilities such as software bugs or incorrect system administration are often exploited by malicious users to intrude into target systems. An intrusion detection system (IDS) is a defense system, which detects hostile activities in the protected network. The key is then to detect and possibly prevent activities that may compromise system security, or a hacking attempt in progress including reconnaissance and/or data collection phases that involve subsequent attacks. Intrusion is a series of concatenated activities that pose threat to the safety of IT resources from unauthorized access to a specific computer or address domain on the part of authorized users or outsiders.

The primary assumptions the intrusion detection is based on are the system activities are observable and normal and intrusive activities have distinct evidence [4]. Based on the intrusion detection method IDS can be categorized into two main categories: misuse based and anomaly based [13]. Misuse based IDS, also referred to as signature based IDS, act similar to virus scanners and look for attack signatures or selected text strings [26, 35]. Since any action that is not clearly considered prohibited is allowed, their accuracy is very high, but they do not achieve completeness to disclose novel attacks or variations of familiar attacks.

Anomaly-based IDS detect the computer intrusions by monitoring system activity and classifying it as either *normal* or *anomalous*. They rely on the assumption that each intrusion will reflect some deviations from normal system activity [1]. These systems construct profiles that represent normal usage and then use current behavior data to detect a possible mismatch between profiles and recognize possible attack attempts.

In order to match event profiles, the IDS is required to produce initial user profiles to train the system with regard to legitimate user behaviors. Then the system uses these profiles as a baseline describing normal or expected user activity. It detects intrusion by observing a deviation from the normal behavior of the system or the users [33].

The main advantage of anomaly based IDS is the potential to detect novel attacks or unknown attacks, as well as attempts to exploit new and unforeseen vulnerabilities regardless of whether the source is a privileged internal user or an unauthorized external user. Anomaly detection approaches have the additional advantage that learning to describe normal activity can be automated. Another benefit of this approach is the less dependence of the IDS on operating environment and the ability to detect abuse of user privileges.

In order to determine what is attack traffic, the system must be taught to recognise normal system activity [32]. This task can be accomplished in several ways. Denning [8] describes an approach that builds profiles based on login times and resources (e.g. files, programs) that users access. Simple statistical methods are used to determine whether observed user behavior conforms to the stored model. Unfortunately, this behavior can suddenly change and is usually not well predictable. As a consequence the focus was shifted from user to program behavior. The execution of a program is modeled as a set of system call

sequences [11], which occur during 'normal' program execution. When the observed sequences deviate from the expected behavior the program is assumed to perform something unintended, possibly because of a successful attack.

There are different approaches to describe normal user activity and the deviations from this baseline – finite automata [15, 23], machine learning [24, 34], neural networks [7, 36], Hidden Markov Model [21, 25, 27], genetic algorithms [9, 14], wavelet analysis [10, 30], statistics [18, 19], etc. The present paper focuses its considerations on the anomaly detection at application level, introduced by [12]. They examined the short sequences of system call traces produced by the execution of the privileged programs at Sun OS system. These processes are of special interest of the attacker, as they run with administrative rights and have access to system resources that are inaccessible for ordinary users.

This paper proposes a method of intrusion detection identification, which is based on a preliminarily composed database of normal user activity patterns. The main purpose is to examine the current user activity and to calculate the probability the current activity to be normal user activity, using the Hidden Markov Model (HMM) [28, 29] – a powerful finite state machine, expedient for various types of pattern recognition problems. The proposed methodology is applied for the attacks detection during the normal activities in the system. As our goal is to distinguish the normal activity patterns from abnormal ones, we consider this detection as decoding problem.

The present method consists of two stages – the first contains the HMM initial creation and its adjustment using the gradient method, and the second one includes the intrusion recognition using the BCJR decoding algorithm [2]. In Section 2 we review some of the standard facts on these techniques. Some of our experimental results are provided in Section 3, including the experimental data, the results of the intrusion detection itself, as well as some results analysis and errors evaluation.

II. Outline of the Methodology

A. The system model

Let's consider a HMM with N states: S_1, S_2, ..., S_N which the system passes through its work in discrete moments of time $t=1, 2, .. ,T,...,$ and that the probability of occupying a state is determined solely by the preceding state. Let $O=(O_1, O_2, ... O_T)$ is the observation sequence at the moments $t=1, 2, ..., T,$ where each O_t is a certain element $v_k \in V$, where the set V is the observations set with M elements in number. We denote the state sequence of HMM at the moments $t=1, 2, ..., T$ with $Q=(q_1, q_2, ..., q_T)$. HMM is completely specified by the ordered triple $\lambda = (A, B, \pi)$:

- The vector π is the initial probability distribution $\pi = (\pi_1, \pi_2, ..., \pi_N)$ for the HMM states.
- The state transition probability matrix $A=\{a_{ij}, 1 \le i \le N, 1 \le j \le N\}$, $0 \le a_{ij} \le 1$ and $\sum_{j=1}^{N} a_{ij} = 1$ is a square matrix with the elements which represent the probability of transitioning from given state to another possible state.
- The observation probability distribution is a non-square matrix $B=\{b_j(O_k), 1 \le j \le N, 1 \le k \le M\}$, with dimensions number of states by number of observations. It represents the probability that a given observable symbol will be emitted by a given state.

We consider those processes only in which the state transition probabilities do not change with time, i.e. $P(q_t = S_j \mid q_{t-1} = S_i) = a_{ij}$ the probability of transiting from state S_i to state S_j does not depend on the moment of time t (stationarity assumption).

The main goal for the HMM is to describe the system behavior during specific period of time. We create an initial HMM, which is tuned up using the learning problem in order to achieve this goal.

This adjustment is performed by determination of the model parameters A, B and π for given HMM λ in order to maximize $L = P(O|\lambda)$ for the observation sequence O. This problem is known as learning problem. There are several optimization criteria for learning, out of which a suitable one is selected depending on the application. We apply the Maximum Likelihood (ML) as optimization criteria in the present paper.

B. ML criterion

In ML we try to maximize the probability of a given sequence of observations O, belonging to a given class w, given the HMM $\lambda = (A, B, \pi)$ of the class w, with respect to the parameters of the model $\lambda = (A, B, \pi)$. There is no known way to analytically determine the model $\lambda = (A, B, \pi)$, which maximize the quantity $L = P(O|\lambda)$. But we can choose model parameters such that it is locally maximized, using an iterative procedure, like a gradient based method. In this method any parameter Θ of the HMM λ is updated according to the standard formula

$$\Theta_{new} = \Theta_{old} - \eta \left[\frac{\partial J}{\partial \Theta} \right]_{\Theta=\Theta_{old}}, \quad (1)$$

where J is a quantity to be minimized. In our case we set $J = -\log p(O|\lambda) = -\log L$. The minimization of J is equivalent to the maximization of L. We have

$$L = \sum_{i=1}^{N} p(O, q_t = i|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i), \quad (2)$$

where:

- the forward variable
$$\alpha_t(i) = p(O_1, O_2, ..., O_t, q_t = S_i \mid \lambda)$$
is defined as the probability of the partial observation

sequence $O_1, O_2, ..., O_t$, when it terminates at the state s_i and can be calculated using the following recursive steps:

$$\alpha_1(j) = \pi_j b_j(O_1),\ 1 \le j \le N$$

$$\alpha_{t+1}(j) = b_j(O_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij},\ 1 \le j \le N,\ 1 \le t \le T-1 \quad (3)$$

- the backward variable

$$\beta_t(i) = p(O_{t+1}, O_{t+2}, ..., O_T, q_t = S_i \mid \lambda)$$

can be defined as the probability of the partial observation sequence $O_{t+1}, O_{t+2}, ..., O_T$, given that the current state is S_i and as in the case of $\alpha_t(i)$ there is a recursive relationship which can be used to calculate $\beta_t(i)$ efficiently:

$$\beta_T(i) = 1,\ 1 \le i \le N$$

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(O_{t+1}),\ 1 \le i \le N,\ 1 \le t \le T-1 \quad (4)$$

Differentiating J with respect to an arbitrary parameter Θ

$$\frac{\partial J}{\partial \Theta} = -\frac{1}{L} \frac{\partial L}{\partial \Theta}.$$

Since there are two main parameter sets in the HMM, transition probabilities a_{ij} and observation probabilities $b_j(O_k)$, we can find the derivative $\frac{\partial L}{\partial \Theta}$ for each of the parameter sets and hence the gradient $\frac{\partial J}{\partial \Theta}$.

1. Gradient with respect to transition probabilities. Using the chain rule

$$\frac{\partial L}{\partial a_{ij}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \alpha_t(j)} \frac{\partial \alpha_t(j)}{\partial a_{ij}}.$$

By differentiating (2) with respect to $\alpha_t(j)$ we get $\frac{\partial L}{\partial \alpha_t(j)} = \beta_t(j)$ and differentiating (3) with respect to a_{ij} we obtain $\frac{\partial \alpha_t(j)}{\partial a_{ij}} = b_j(O_t)\alpha_{t-1}(i)$. Then

$$\frac{\partial J}{\partial a_{ij}} = -\frac{1}{L} \sum_{t=1}^{T} \beta_t(j) b_j(O_t) \alpha_{t-1}(i).$$

2. Gradient with respect to observation probabilities. Using the chain rule

$$\frac{\partial L}{\partial b_j(O_t)} = \frac{\partial L}{\partial \alpha_t(j)} \frac{\partial \alpha_t(j)}{\partial b_j(O_t)}.$$

By differentiating (3) with respect to $b_j(O_t)$ we get $\frac{\partial \alpha_t(j)}{\partial b_j(O_t)} = \frac{\alpha_t(j)}{b_j(O_t)}$. Then

$$\frac{\partial J}{\partial a_{ij}} = -\frac{1}{L} \frac{\alpha_t(j)\beta_t(j)}{b_j(O_t)}.$$

The attacks recognition can be considered as a decoding problem. One fundamental decoding principle is symbolwise maximum a posteriori decoding – the concept of optimally decoding each symbol. As an example of this decoding we applied BCJR decoding algorithm. Our second step is to use this algorithm to estimate random parameters with prior distributions. The algorithm scans the traces of the system activity and compares with the patterns of normal user activity.

C. The BCJR algorithm

The description of the BCJR algorithm can be done based on log-likelihood ratios (LLR). The LLR are represented as follows

$$\Lambda_i = \ln \frac{P(m_i = 1 | O_i)}{P(m_i = 0 | O_i)}.$$

where m_i is the message bit associated with the state transition q_i to q_{i+1} and $P(m_i = 1 | O_i)$ is the a posteriory probability (APP) in which the bit, determining the presence of attack, is equal to 1. If the LLR of an observation is positive, it implies that m_i is most likely to be a 1 and if it is negative, m_i is most likely to be a zero. We can express $P(m_i = 1 | O_i)$ as follows:

$$P(m_i = 1 | O_i) = \sum_{S_1} P(s_i \to s_{i+1} | O_i) = \sum_{S_1} \frac{P(s_i \to s_{i+1}, O_i)}{P(O_i)}$$

,

where S_1 represents the set of all state transitions for which the input symbol is O_i. Similarly,

$$P(m_i = 0 | O_i) = \sum_{S_0} P(s_i \to s_{i+1} | O_i) = \sum_{S_0} \frac{P(s_i \to s_{i+1}, O_i)}{P(O_i)}$$

,

where S_0 is the set of all pairs of states which transient from a state s_i at time i to a state s_{i+1} at time $i+1$ under the input symbol not O_i. Hence, the LLR of the i^{th} observation is obtained as:

$$\Lambda_i = \ln \frac{\sum_{S_1} P(s_i \to s_{i+1}, O_i)}{\sum_{S_0} P(s_i \to s_{i+1}, O_i)}.$$

We partition the joint probability of $P(s_i \to s_{i+1}, O_i)$ into three parts using Bayes' rule:

$$P(s_i \to s_{i+1}, O_i) = \alpha(s_i)\gamma(s_i \to s_{i+1})\beta(s_{i+1}),$$

where $\alpha(s_i) = P(s_i, O_{-1})$ is the forward variable, $\beta(s_{i+1}) = P(O_{i+1} | s_{i+1})$ is the backward variable and

$$\gamma(s_i \to s_{i+1}) = P(s_{i+1}, O_i | s_i)$$

represents the probability of the state transition from s_i to s_{i+1}, given the current state is s_i which is called the Branch metric associated with the state transition $s_i \to s_{i+1}$. The Branch metric can be expressed as:

$$\gamma(s_i \to s_{i+1}) = P(s_{i+1}|s_i)P(O_i|s_i \to s_{i+1}) = P(m_i)P(O_i|x_i)$$

The first term $P(m_i)$ represents the a priori information.

The second term $P(O_i|x_i)$ is straight-forward.

$$P(O_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(O_i - ax_i)^2}{2\sigma^2}\right],$$

where x_i is the result of the transition from the state s_i to state s_{i+1} during the normal system activity.

III. Simulation Experiments

A. The Experiment Data

Following the scheme of intrusion detection described in section II, we have conducted several experiments. The experimental data were obtained from a project performed by the researches in the Computer Science Department, University of New Mexico [37].

The data are obtained from Unix and Sun SPARCstations system examination during some period of time and consist of normal user activity patterns of some privileged processes executed with administrative rights as well some anomalous data. The privileged processes are of special interest of the attacker as they perform some services which require access to system resources that are inaccessible to ordinary users. The methods for pattern generation are described in [11] and [12]. They substantiate the short sequences of system calls are reliable discriminator between normal and anomalous activities in the system. Each pattern is a sequence of system calls, which are the results of the examined process. The input data files are sequences of ordered pairs of numbers, where each line consists of one pair. The first number in each pair is the process ID (PID) of the process executed, and the second one is the system call number. Forks are taken into account as separate processes and their execution results are considered as normal user activity. Table I contains some examples of input data.

The experimental data include normal user activity traces as well as intrusion data. The normal activity patterns compose the set of the states *S* and the intrusion activity patterns compose the set *V*. The initial system model was created according to these data. Then the model is tuned using the gradient method, described in section II.B. The obtained model is used as a database describing normal system activity. Then the BCJR algorithm was applied in order to distinguish normal traces from abnormal ones. As a result of BCJR algorithm we obtain the LLRs which represent the probability of intrusion occurrence at the given moment of time.

TABLE I.
SYSTEM CALL DATA, CONTAINING PID AND SYSTEM CALL NUMBER

PID	1393	1393	...	1423
System calls	112	19	...	105

TABLE II.
NUMBERS OF ITERATIONS AND THE VALUES OF *L* DEPENDING ON THE VALUES OF η WHEN *T*=10

T=10	Number of iterations	L
η=0,0001	322	4.20353e-13
η=0,0002	161	3.41980e-13
η=0,0003	107	3.21028e-13
η=0,0004	80	7.60207e-14
η=0,0005	64	1.15635e-13
η=0,00001	3224	3.94035e-13
η=0,00002	1611	3.78713e-13
η=0,00003	1074	3.87351e-13
η=0,00004	805	3.58734e-13
η=0,00005	644	3.58219e-13

TABLE III.
NUMBERS OF ITERATIONS AND THE VALUES OF *L* DEPENDING ON THE VALUES OF η WHEN *T*=15

T=15	Number of iterations	L
η=0,0001	322	2.88298e-17
η=0,0002	162	1.88268e-17
η=0,0003	108	4.77443e-18
η=0,0004	82	8.53916e-18
η=0,0005	65	5.47520e-19
η=0,00001	3207	2.54814e-17
η=0,00002	1604	2.47694e-17
η=0,00003	1070	2.46345e-17
η=0,00004	802	2.22364e-17
η=0,00005	642	2.21205e-17

We used a slide window with length *T* to cross the traces of current user activity, i.e. the system observations, which compose the set *O*. We accomplished experiments with the following values of *T*: 10, 15 and 20. Given an unknown observation sequence, the ML-criterion finds the model λ which maximizes the value of $L = P(O|\lambda)$. For standard gradient descent we use learning rate η from 0,000001 to 0,000009 with step 0,000001, from 0,00001 to 0,00009 with step 0,00001 and from 0,0001 to 0,0009 with step 0,0001 for both observation and transition probabilities. Initially we examined the data about the process synthetic sendmail. Some of the obtained results are summarized in Tables II and III which represent the number of iterations and the value of *L* for η=0,0001-0,0005 and η=0,00001-0,00005 when we examine 10 or 15 unknown observations.

The algorithm exhibits a tendency of growth of the number of iterations when we increase the number of observations and decrease the learning rate η. The number of iterations necessary for the model training is similar when *T*=10 and 15. One of the greatest problems in training large models with gradient descent is to find an optimal learning rate. A small one will slow down the speed and significantly increase the number of iterations. On the other hand, a large one will probably cause oscillations during training and finally leading to no useful model would be trained.

B. The Detection Results

Anomalous data was examined using the BCJR decoding algorithm which compares the traces of the system activity for *T*=10, 15 and 20 with the patterns of

normal user activity. The intrusion detection problem is considered as a decoding problem. The results of the algorithm are the values of LLR, where each LLR is the logarithmic ratio of the probability of attack presence and the probability of normal activity at specific moment *t*. We assume that the values of LLR greater than 0 denote an attack presence.

Figure 1 contains the values of LLRs when *T*=10 for values of η=0,00006-0,00009. As we can see the algorithm recognizes the observation O_6 as anomalous regardless of the value of η. So the chosen values of η do not have significant influence on the method ability for attacks detection.

Some of the results for *T*=15 and 20 and η=0,00006-0,00009 are represented at Figure 2 and 3. The positive values of LLR stand for observations considered as attacks. For instance, from Figure 3 IDS for *T*=20 the method most likely checks O_6, O_8 , O_{14}, O_{19}, O_{20} as attacks.

Figures 4 and 5 contain the comparison between the values of LLR's corresponding to the same observations with different values of *T*. For instance, from Figure 4 we see that the IDS for *T*=15 most likely checks O_{12} as an attack while the IDS for *T*=20 checks it as normal system activities and IDS for *T*=20 most likely checks O_8 and O_{14} as attacks while IDS for *T*=15 checks these observations as normal system activities. We assume the results for the greater value of *T* are more reliable due to the larger number of observations considered in LLR's calculation.

Some of the results for *T*=10, η=0,00004-0,00006 and the processes: synthetic ftp, named and xlock are summarized at Figures 6, 7 and 8.

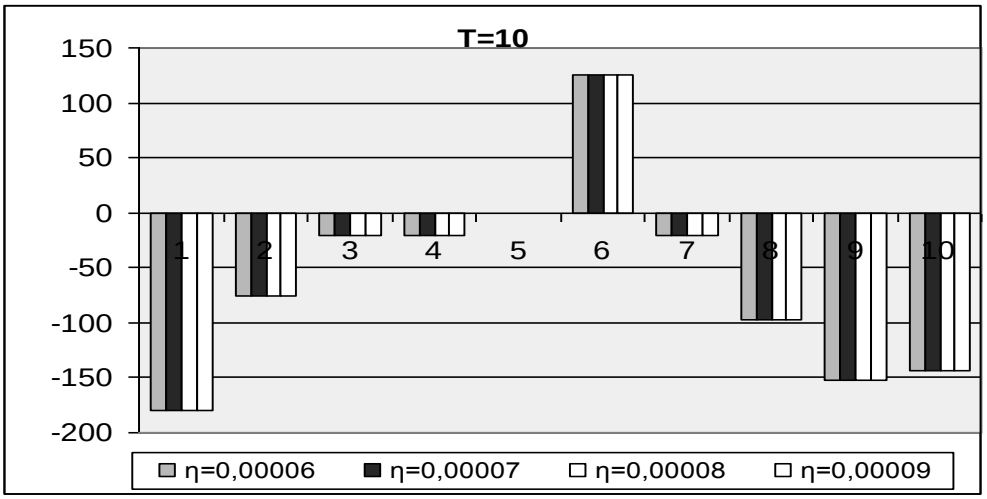


Figure 1. Values of LLR depending on the value of η when *T*=10.

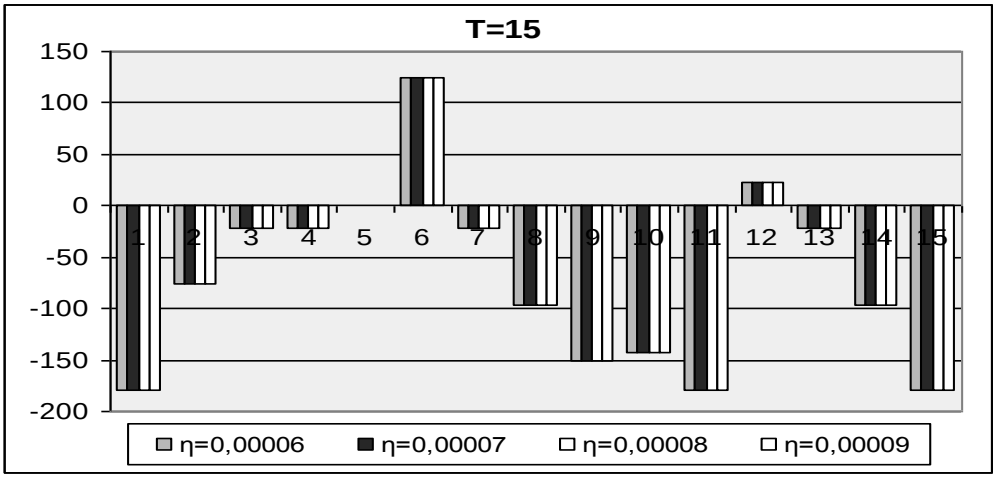


Figure 2. Values of LLR's depending on the values of η when T=15

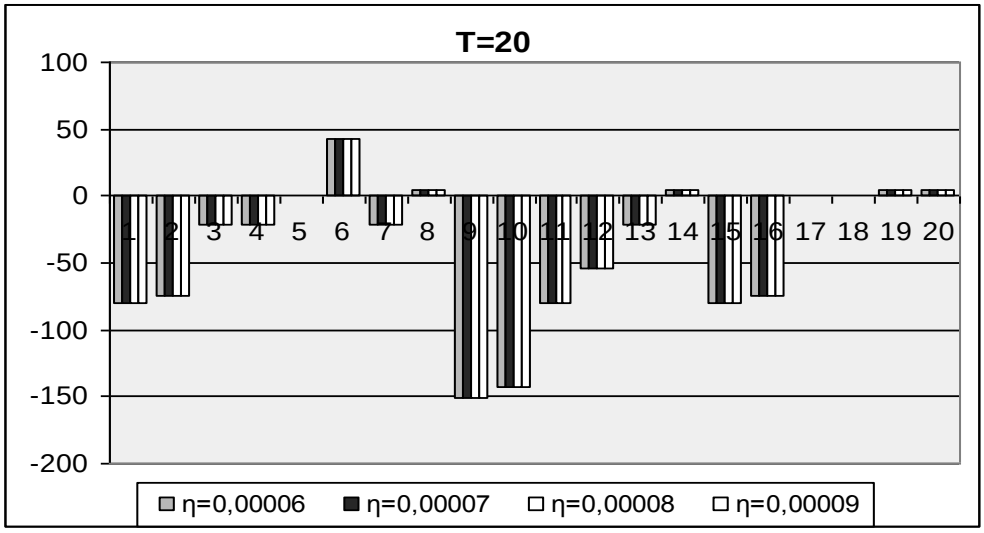
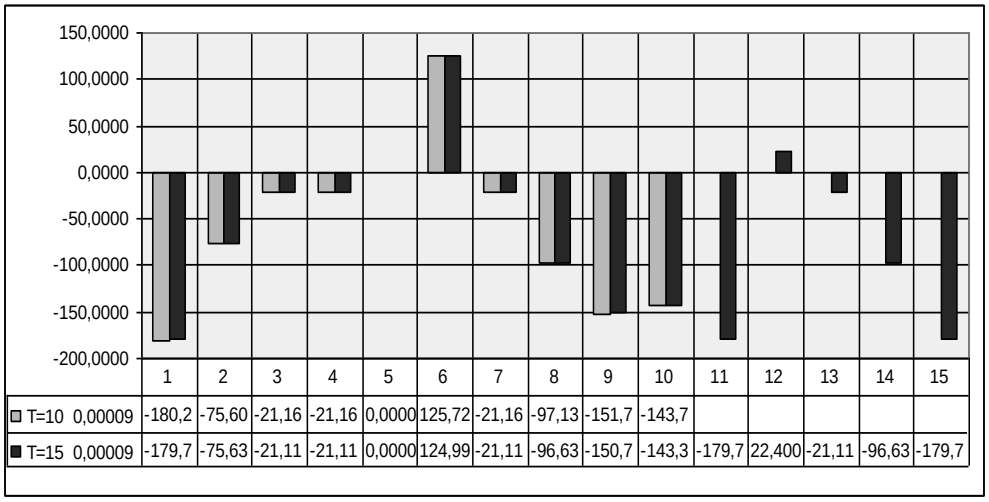


Figure 3. Values of LLR's depending on the values of η when T=20



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T=10 0,00009	-180,2	-75,60	-21,16	-21,16	0,0000	125,72	-21,16	-97,13	-151,7	-143,7					
T=15 0,00009	-179,7	-75,63	-21,11	-21,11	0,0000	124,99	-21,11	-96,63	-150,7	-143,3	-179,7	22,400	-21,11	-96,63	-179,7

Figure 4. Values of LLR for T=10 and T=15 and η=0.00009

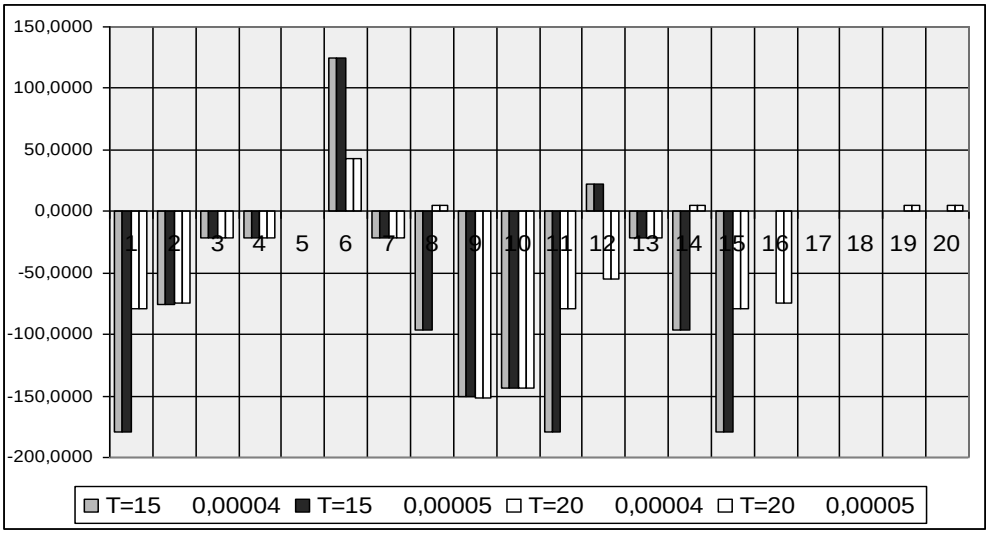


Figure 5. Values of LLR for T=15 and T=20 and η=0.00004 and η=0.00005.

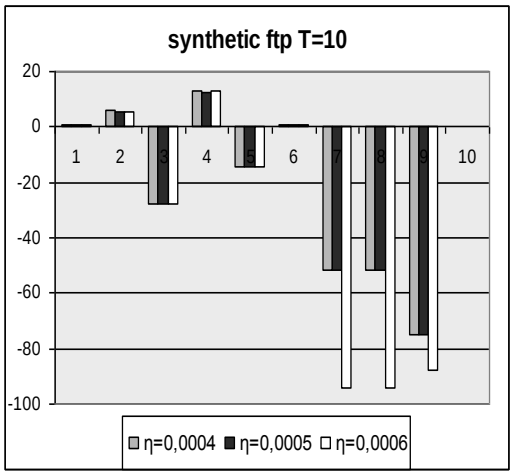


Figure 6. Values of LLR depending on the value of η when T=10 for synthetic ftp.

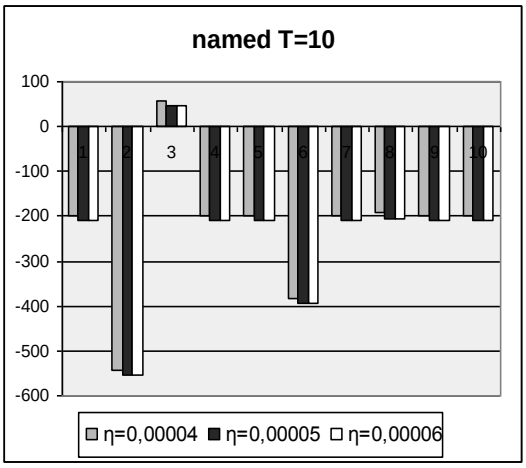


Figure 7. Values of LLR depending on the value of η when T=10 for named.

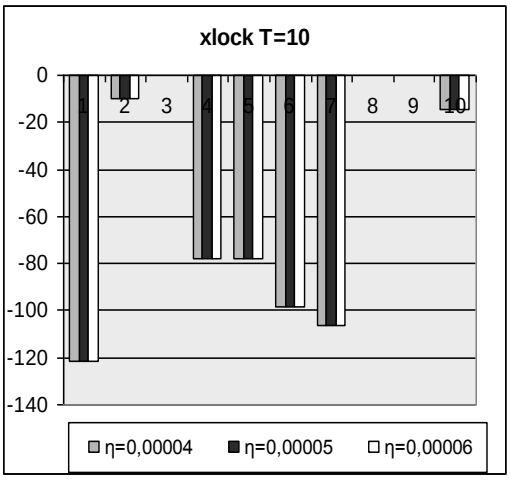


Figure 8. Values of LLR depending on the value of η when T=10 for xlock.

C. The Results Analysis

The false positive rate (FPR) is the frequency with which the IDS reports malicious activity in error. The true danger of a high false positive rate lies in fact that it may cause to ignore the system's output when legitimate alerts are raised. The false negative rate (FNR) is the frequency with which the IDS fails to raise an alert when malicious activity actually occurs, i.e. it represents the undetected attacks on a system. False negative rate changes in an inverse proportion to false positive rate.

In order to evaluate the false positives rate we applied a method used by Hoang et.al. [16]. This approach is based on the assumption that as a normal trace sequence does not contain any intrusions, any reported alarms could be considered as false alarms. From the normal trace we generated a list of *n* consecutive short sequences of system calls, using a sliding window of length *T*. Then, each short sequence of the list is evaluated by the detection method to determine if it is normal or abnormal. We counted all abnormal sequences for the whole list as *m*. The false positive rate is calculated as *m/n*. Figures 9, 10 and 11 represent some values of LLR for the processes: synthetic ftp, named and xlock, which are used to determine what short sequence of the list is normal or abnormal.

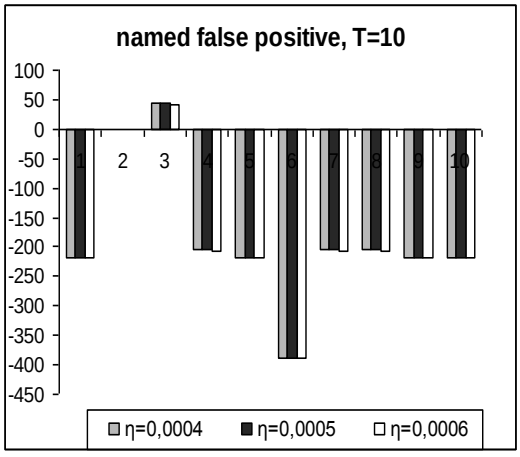


Figure 9. Values of LLR representing false positives for named.

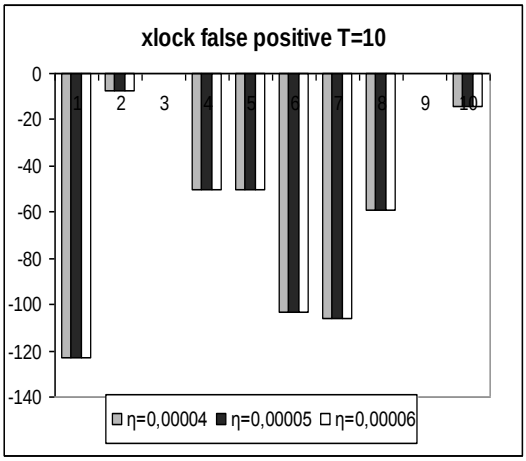


Figure 10. Values of LLR representing false positives for xlock.

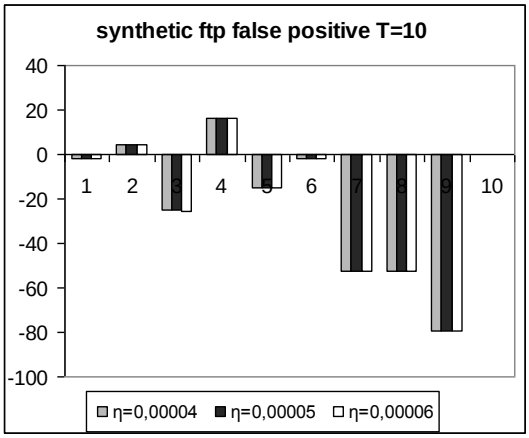


Figure 11. Values of LLR representing false positives for synthetic ftp.

Table IV contains the values of the false positive rate, the false negative rate and the accuracy for processes synthetic ftp, named and xlock.

Figures 12, 13 and 14 contain graphs of the false negative rate, fraction of intrusions incorrectly not detected, and the false positive rate, fraction of non-intrusions incorrectly detected, for the same input.

Another statistical method for evaluation the IDS effectiveness is calculation of sensitivity and specificity. Sensitivity is defined as the true positive rate (TPR), i.e. intrusion correctly detected. Mathematically, sensitivity is expressed as follows:

$$\frac{\text{True Positive Rate}}{\text{True Positive Rate + False Negative Rate}}$$

TABLE IV.
THE FALSE ALARMS RATE AND THE ALGORITHM ACCURACY

Process	False positive rate	False negative rate	Accuracy
synthetic ftp	17%	39%	72%
synthetic sendmail	11%	33%	83%
named	5%	23%	86%
xlock	3%	15%	91%

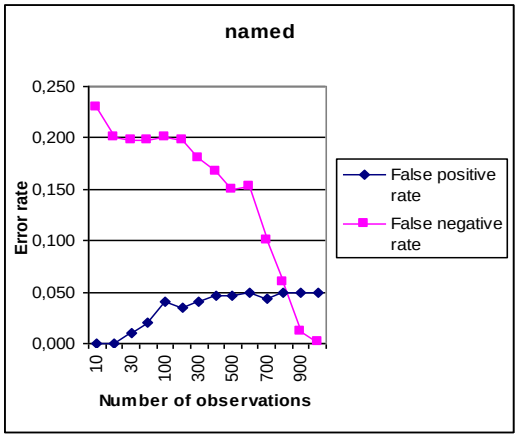


Figure 12. Error rate for named.

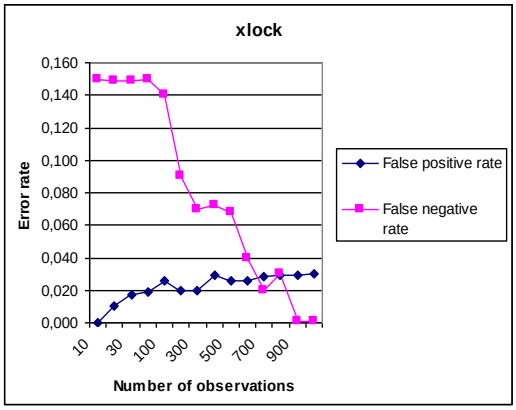


Figure 13. Error rate for xlock.

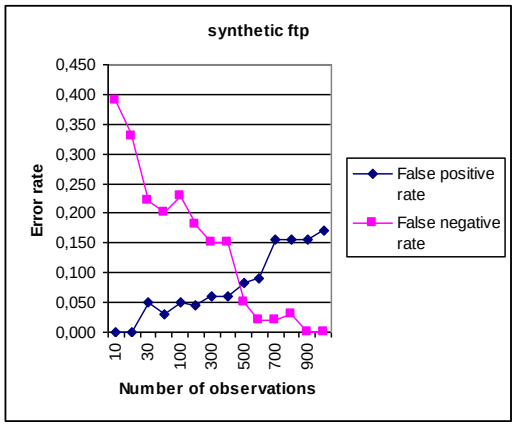


Figure 14. Error rate for synthetic ftp.

The false negative rate is equal to one minus the sensitivity. True negative rate (TNR) represents an IDS that is correctly reporting that there are no intrusions.

TABLE V.
THE CROSSOVER ERROR RATE

Process	CER
synthetic ftp	0,07
inetd	0,06
named	0,05
xlock	0,03
stide	0,05

Specificity is expressed as

$$\frac{\text{True Negative Rate}}{\text{True Negative Rate + False Positive Rate}}$$

The false positive rate is equal to one minus the specificity. Sensitivity and specificity for the processes inetd and stide are respectively 0,87; 0,93 and 0,92; 0,89.

The crossover error rate (CER) is defined as adjusting the system's sensitivity until the false positive rate and the false negative rate are equal. The crossover error rate for considered processes is represented in Table V. In order to achieve a balance between false positive rate and false negative rate, we may select the IDS with the lowest crossover error rate.

The receiver operating characteristic (ROC) curve is a

method of graphically demonstrating the relationship between sensitivity and specificity. An ROC space is defined by FPR and TPR as *x* and *y* respectively, which depicts relative trade-offs between true positive and false positive. The decision threshold divides the normal activities into a true negative and a false positive group, and the attack sequences into a true positive and a false negative group. As the decision threshold moves to the right along the *x*-axis, sensitivity ranges from one, when all tests are read as abnormal (no false negatives), to 0, when all are normal (no true positives). Maximal sensitivity is realized when all tests are reported as abnormal. Specificity moves in concert from 0 (no true negatives) to 1 (no false positives). Maximal specificity is achieved by reporting all tests as normal. The best possible prediction method would yield a point in upper left corner (0,1) of the ROC space, representing 100% sensitivity (all true positives are found) and 100% specificity (no false positives are found). This point is called a perfect classification. The diagonal line (from the left bottom to the right corner) divides the ROC space in areas of good and bad classification. Points above this line indicate good classification results; while points below the line indicate wrong results (see Figures 15 and 16). Each sensitive value can be plotted against its corresponding specificity value to create the diagram for the processes inetd and stide in Figures 17 and 18 respectively.

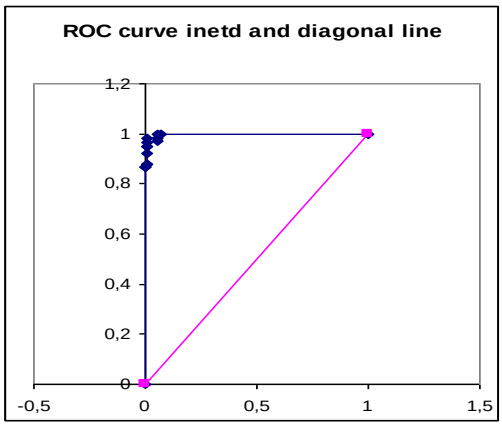


Figure 15. The ROC curve and the diagonal line for inetd

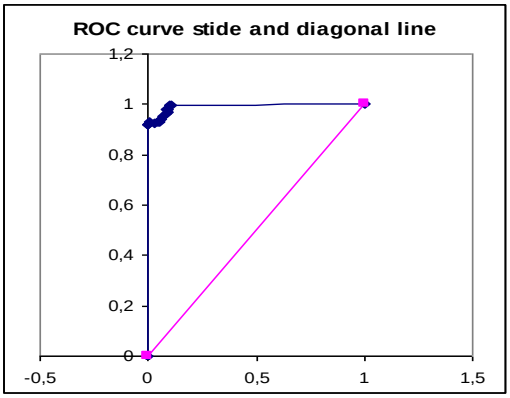


Figure 16. The ROC curve and the diagonal line for stide

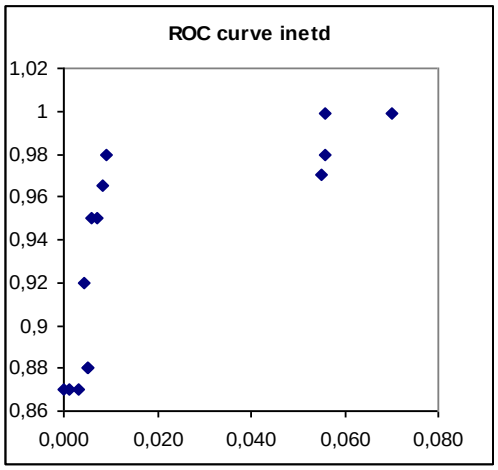


Figure 17. The ROC curve for inetd

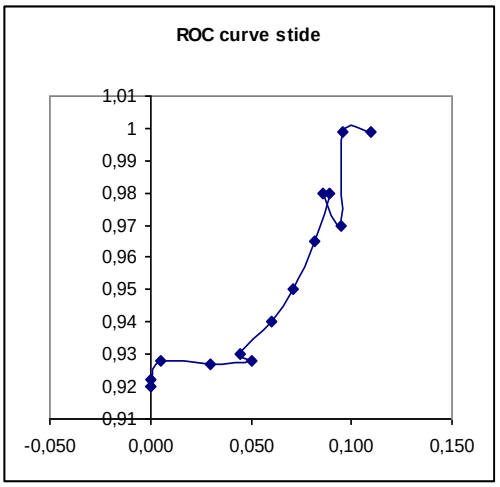


Figure 18. The ROC curve for stide

IV. DISCUSSIONS

Based on simulation results (low crossover error rate and false alarm rate, and satisfactory level of accuracy), the application of the proposed method for anomaly-based intrusion detection is technically feasible.

An advantage of the described probabilistic method is its potential to detect an unknown attack the first time it appears, as it is based on the BCJR decoding algorithm. As a result of the algorithm we obtain the probability of attack presence, divided by the probability of attack absence instead of one-to-one mapping between the current patterns and those in the database. We applied a model training using the learning problem based on HMMs in order to learn normal and abnormal patterns of program behavior from its execution trace to generalize upon the method introduced in [11]. The proposed method is shown to be able to accurately detect anomalous intrusions. It not only increases the level of confidence but improves the false alarm and detection rates also. Our experiments demonstrate that learning problem combined with the BCJR decoding algorithm can indeed play an important role in intrusion detection of computer systems.

Nevertheless, there are some open issues, which warrant further attention. A disadvantage of the described method is its considerable price, as the BCJR algorithm has $O(TN^2)$ complexity. As N is the number of the states, i.e. the number of normal user activity patterns in the database, its value could be significant in the case of a large system. Another disadvantage of the anomaly based IDS in general is the creation of the database containing the user profiles, which could be a task of considerable difficulty, especially during the ML training. The gradient method has $O(TN^2)$ complexity at each training step. We should mention this training is accomplished once before the observation decoding which is a continuous process during the system work.

Our method is based on the definition of normal behavior in terms of short sequences of system calls, described by Forrest et. al. ([11] and [12]). With the purpose of simplicity, this method ignores the parameters passed to the system calls, and look only at their temporal orderings. We should mention this definition of normal behavior ignores many other important aspects of process behavior, such as timing information instruction sequences between system calls, and interactions with other processes.

V. Conclusion

The intrusion detection is beginning to assume enormous importance in today's highly connected network environment. The combination of facts such as the unbridled growth of the Internet and the vast financial possibilities opening up in electronic trade makes it an important field of research.

Hidden Markov Methodology, with particular care to the parameter estimation and the training phase, represents a powerful approach for creating anomaly detection method which can find whether the traffic is normal or containing some sort of anomaly. This paper investigated the capabilities of this methodology in anomaly detection method. The model training is performed using ML criterion, based on the gradient method. Since we considered the attacks recognition as a decoding problem, we applied the BCJR algorithm combined with gradient based method. The training of HMM model is expensive but the detection of the attacks is more efficient. This results in a system that would be able to accurately detect the intrusions. Although the proposed model is a host-based intrusion detection scheme, it has the potential for use in networked environments.

References

[1] Abraham, A., Thomas, J., Distributed Intrusion Detection Systems: A Computational Intelligence Approach, *Applications of Information Systems to Homeland Security and Defense*, Idea Group Inc. Publishers, USA, Chapter 5, pp. 105-135, 2005.

[2] Bahl L., Jelinek J., Raviv J., Raviv F., Optimal Decoding of Linear Codes for minimizing symbol error rate, *IEEE Transactions on Information Theory*, vol. IT-20, March 1974, pp.284-287.

[3] Chan P., M. Mahoney, M. Arshad, Learning Rules and Clusters for Network Anomaly Detection, *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, George Mason University, Technical Report CS-2003-06, 2003.

[4] Chebrolu, S.; Abraham A.; Thomas, J. P., Feature deduction and ensemble design of intrusion detection systems, *Computers & Security*, Volume 24, Issue 4, 1 June 2005, pp. 295-307.

[5] Christodoresku M., S. Jha, Static Analysis of Executables to Detect Malicious Patterns, *In Proceedings of the 12th USENIX Security Symposium*, August 2003, pp.169—186.

[6] Crosbie M., G. Spafford, Applying Genetic Programming to Intrusion Detection, *Working Notes for the {AAAI} Symposium on Genetic Programming*, 1995, pp.1-8.

[7] Degang Y., C. Guo, W. Hui, L. Xiaofeng, Learning vector quantization neural network method for network intrusion detection, *Wuhan University Journal of Natural Sciences*, Volume 12, Number 1, 2007, ISSN 1007-1202, pp. 147-150.

[8] Denning D., An intrusion-detection model, *In Proceedings of IEEE Symposium on Security and Privacy*, Oakland, USA, 1986, pp. 118-131.

[9] Diaz-Gomez P. A., D. F. Hougen, MISUSE DETECTION: A Neural Network vs. a Genetic Algorithm Approach, *In Proceedings of the Nineth International Conference on Enterprise Information Systems*, 2007, pp. 459-462.

[10] Ezekiel S., W. Oblitey, R. Trimble, Network Signal Analysis: A Wavelet Approach, *Parallel and Distributed Computing and Networks*, Innsbruck, Austria, 2005, 456-228, ISBN: 0-88986-468-3.

[11] Forrest S., S.A. Hofmeyr, A. Somayaji, T.A. Longtaff, A Sense of Self for Unix Processes. *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitors, CA, pp.120-128.

[12] Forrest S., S.A. Hofmeyr, A. Somayaji, Intrusion detection using sequences of system calls, *Journal of Computer Security*, Vol. 6, 1998, pp. 151-180.

[13] Fuchsberger A., Intrusion Detection Systems and Intrusion Prevention Systems, *Information Security Technical Report*, Volume 10, Issue 3, 2005, pp. 134-139.

[14] Haghighat A. T., M. Esmaeili, A. Saremi, V. R. Mousavi, "Intrusion Detection via Fuzzy-Genetic Algorithm Combination with Evolutionary Algorithms," ICIS, pp. 587-591, *6th IEEE/ACIS International Conference on Computer and Information Science* (ICIS 2007), 2007.

[15] Han Z.F.; J.P. Zou; H. Jin; Y.P. Yang; J. H Sun, Intrusion detection using adaptive time-dependent finite automata, *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 2004, Vol. 5, pp. 3040 – 3045.

[16] Hoang X.D., J. Hu, P. Bertok, A Multi-layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls, *11th IEEE International Conference on Networks* (ICON 2003), Sydney, Australia, 2003.

[17] Hou H., G. Dozier, Immunity-based intrusion detection system design, vulnerability analysis, and GENERTIA's genetic arms, *Symposium on Applied Computing Proceedings of the 2005 ACM symposium on Applied computing*, ISBN:1-58113-964-0, 2005, pp. 952 – 956.

[18] Jin S., D. S. Yeung, X. Wang, Network intrusion detection in covariance feature space, *Pattern Recognition*, Volume 40, Issue 8, 2007, pp. 2185-2197.

[19] Katos V., Network intrusion detection: Evaluating cluster, discriminant, and logit analysis, *Information Sciences*, Volume 177, Issue 15, 2007, ISSN: 0020-0255, pp. 3060-3073.

[20] Kemp M., For whom the bells toll: effective IDS deployment strategies, *Network Security*, Volume 2005, Issue 5, May 2005, pp. 16-18.
[21] Khanna R., H. Liu, Distributed and control theoretic approach to intrusion detection, *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, Honolulu, Hawaii, USA, ISBN: 978-1-59593-695-0, pp. 115 - 120.
[22] Ko C, G. Fink, K Levitt. Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring, *Proceedings of the 10th Annual computer Security Applications Conference*, pp.134-144, 1994.
[23] Michael C. C., A. Ghosh, "Simple, state-based approaches to program-based anomaly detection", *ACM Transactions on Information and System Security*, Vol. 5, No. 3, August 2002, pp. 203-237.
[24] Moskovitch, R. P., S. Gus, I. Stopel, D. Feher, C. Parmet, Y. Shahar, Y. Elovici, Host Based Intrusion Detection using Machine Learning, *Intelligence and Security Informatics*, 2007 IEEE, USA, E-ISBN: 1-4244-1329-X, pp. 107-114.
[25] Ourston, D.; Matzner, S.; Stump, W.; Hopkins, B. Applications of hidden Markov models to detecting multi-stage network attacks, *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003. Issue 6-9, pp.10-15.
[26] Platt A., N. Goharian: Short Query Sequences in Misuse Detection, *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, ISI 2007, New Brunswick, New Jersey, USA,. IEEE 2007, pp.379-382.
[27] Qian Q., M. Xin, Research on Hidden Markov Model for System Call Anomaly Detection, *Intelligence and Security Informatics*, PAISI 2007, Chengdu, China, Volume 4430/2007, pp.152-159
[28] Rabiner L.R., A tutorial on Hidden Markov Models and selected applications in speech recognition, *Proc. IEEE*, 257-286, 77, 2, Feb 1989.
[29] Rabiner L. R., B. H. Juang. An introduction to Hidden Markov Models, *IEEE ASSP Magazine*, pp.4-16, January 1986.
[30] Rawat S., C. S. Sastry, Network Intrusion Detection Using Wavelet Analysis, Intelligent Information Technology, *Proceedings of 7th International Conference on Information Technology*, CIT 2004, Hyderabad, India, ISBN 978-3-540-24126-3, 2004, pp. 224-232.
[31] Reddy Y.B., Genetic Algorithm Approach for Intrusion Detection, *Proceedings of Modeling, Simulation, and Optimization MSO 2004*, Hawaii, ISBN: 0-88986-424-1, 2004.
[32] Sekar R., M. Bendre, Dhurjati, P. Bullineni, "A fast automaton-based method for detecting anomalous program behaviors," *IEEE Symposium on Security and Privacy, 2001*. S&P 2001, pp. 144 – 155.
[33] Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., Zhou S., Specification-based anomaly detection: a new approach for detecting network intrusions, *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 265–274.
[34] Tandon G., P. Chan, On the Learning of System Call Attributes for Host-based Anomaly Detection, *International Journal on Artificial Intelligence Tools*, 15(6), pp. 875-892, 2006.
[35] Tran T.P., T. Jan, A.J. Simmonds, A Multi-Expert Classification Framework for Network Misuse Detection, *From Proceeding (544) Artificial Intelligence and Soft Computing*, ISBN 0-88986-610-4, 2006.
[36] Venkatachalam V., S. Selvan, Intrusion Detection using an Improved Competitive Learning Lamstar Neural Network, *IJCSNS International Journal of Computer Science and Network Security*, VOL.7 No.2, February 2007, pp. 255-263.
[37] University of New Mexico's Computer Immune Systems Project, http://www.cs.unm.edu/~immsec/systemcalls.htm.

Veselina G.Jecheva was in Burgas, Bulgaria in 1971. She received her master degree in Computer Science and Economics from Sofia University "St. Kliment Ohridski" in 1995. She received her PhD degree in Computer Science, especially Information Security from Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, in 2005. She is a lecturer at Burgas Free University, Bulgaria. Her research interests include information security, e-commerce, programming, Web systems.

Evgeniya P. Nikolova was born in Pomorie, Bulgaria, in 1968. Between 1986 and 1991 she studied at Sofia University "St. Kliment Ohridski" and was awarded her master degree in Probability theory and Statistics in 1991. She received her PhD in Informatics (Proper codes) from Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, in 2005. Evgeniya is a lecturer at Burgas Free University, Bulgaria. Her research interests include probability theory and coding theory.