32

New Trends in Network Anomaly Detection

Yasser Yasami[1] and Saadat Pourmozaffari
Amirkabir University of Technology
Iran

1. Introduction

Computer networks are complex interacting systems composed of individual entities such as various devices, workstations and servers. Nowadays, *Internet Protocol* (IP) is used as a dominant layer 3 protocol. The evolving nature of IP networks makes it difficult to fully understand the dynamics of the systems and networks. To obtain a basic understanding of the performance and behavior of these complex networks, large amount of information need to be collected and processed. Often, network performance information is not directly available, and the information obtained must be synthesized to obtain an understanding of the ensemble behavior.

Traditional signature-based intrusion detection techniques use patterns of well-known attacks to match and identify known intrusions. The main drawback of theses techniques is inability to detect the newly invented attacks. To obtain sufficient information about complex network traffic and compensate for the weaknesses of traditional *Intrusion Detection Systems* (IDS), *Anomaly Detection Algorithms* (ADA) are used [G.Maselli & L.Deri, 2003; K. Hwang et al., 2004; A. Lazarevic et al., 2003]. Theses algorithms can be employed as a useful mechanism to analyze network anomalies and detect misbehaviors issued by users, or even unknown signature viruses and worms.

There are two main approaches to study or characterize the ensemble behavior of the network [M. Thottan & C. Ji, 2003]: the first is inference of the overall network behavior and the second is to analyze behavior of the individual entities or nodes. The approaches used to address the anomaly detection problem depend on the nature of the data that is available for the analysis. Network data can be obtained at multiple levels of granularity such as network-level or end-user-level. The methods presented in this chpater are host-based ADA's and are categorized in the latter approach.

In this chapter, we present some ADA's developed based on some classification methods. The goal of this chapter is to classify each user's behavior as anomalous or normal actions in an unsupervised fashion. Four different algorithms are disccusd and compared based on some defined measures.

The experiments are performed on a real evaluation network test bed. Instances are captured in eight consecutive weeks, three weeks of training and five weeks of testing. Some

[1] Yasser Yasami is currently an instructor of computer science & engineering at Payam Nour University.

ARP anomaly criteria are defined. These criteria are applied to the three weeks training instances for generating normal ARP traffic.

Performance evaluation of the approaches is conducted by five performance measures: Sensitivity, Specificity, Negative Likelihood Ratio, Positive Predictive Value, and Negative Predictive Value. Finally some comparisons are performed based on the defined measures.

2. Background and Related Works

Network anomaly detection is a vibrant research area. ARP anomaly detection in particular has been of great interest. Some methods for anomaly detection are based on switch characteristics, such as performance and backplane [D. Ármannsson et al., 2005]. In such methods switch characteristics must be known. Our knowledge is limited to theoretical backplane speed mentioned in datasheets. But, because switch processing power, especially when forwarding and flooding small packets, does not equal to that of theory and performance of switches in high load, small packet traffic degrade dramatically, so using such algorithm, encounters functional limitations.

In other researches [D. Whyte et al., 2005], feature-based approaches for host-based analysis of ARP anomaly detection have been suggested. To achieve more accuracy on the results, more inputs factors to these algorithms are needed to be defined. Furthermore, the proposed factors have correlation with each other. None of these works include any suggestion about correlation between the factors, which affect on their precision.

The proposed algorithm in [Shekhar R. Gaddam et al., 2007] is a supervised ADA. We are not provided with a set of anomalous and normal labeled training instances, mostly. So, supervised algorithms such as the one proposed in [Shekhar R. Gaddam et al., 2007] are confronted with limitations in practical applications. Furthermore, the majority of the works proposed in [N. Ye et al., 2004; D. Mutz et al., 2006; S. Kumar & E.H. Spafford, 1994; C. Kruegel & G. Vigna, 2003] evaluate the performance of anomaly detection methods on the measurements drawn from one application domain, thereby addressing the problem of anomaly detection on limited data instances collected from a single application domain. There are some other approaches that apply machine learning techniques like symbolic dynamics [A. Ray, 2004], multivariate analysis [N. Ye, et al., 2002], neural-networks [Z. Zhang et al., 2001], self-organizing maps [S.T. Sarasamma et al., 2005], fuzzy classifiers [J. Gomez & D.D. Gupta, 2001] and others [H.S. Javitz & A. Valdes, 1991; I. Levin, 2000; D.Y. Yeung & C. Chow, 2002; R. Agarwal & M.V. Joshi, 2000; G. Qu et al., 2005]. Almost all of these anomaly detection approaches apply single machine learning techniques while recent advances in machine learning show that selection, fusion and cascading [A. Verikas et al., 1999; J. Kittler et al., 1998; L.I. Kuncheva, 2002] of multiple machine learning approaches have a better performance yield over individual approaches.

3. Network Anomalies

Network anomalies typically refer to circumstances when network operations deviate from normal network behavior. The anomalies can arise due to various causes such as malfunctioning network devices, bad configuration in network services and operating systems, network overload, malicious denial of service attacks, ill advised applications installed by users, high level users' effort to discover network and gather information about it and its devices, and network intrusions that disrupt the normal delivery of network services. These anomalous events will disrupt the normal behavior of some measurable network data. The definition of normal network behavior for measured network data is dependent on several network specific factors such as dynamics of the network being studied in terms of traffic volume, the type of network data available, and types of applications running on the network. Accurate modeling of normal network behavior is still an active field of research, especially the online modeling of network traffic.

Some of intrusions and malicious usages don't have significant effects on network traffic (i.e. ARP Spoofing). So such misbehavior is not addressed in this chapter. Other types of attacks are based on broadcasting large number of packets with abnormal behavior, as in the case of DoS attacks. Abnormality is generally different from large number of packets, although large number of packets introduces abnormality to network traffic, too. High percentage of packets, degrade network performance. There are other types of attacks which apply broadcast traffic for detecting live hosts in network. Network anomalies can be caused by some unintentional and curious motivations, too. To detect these anomalies an algorithm is introduced in this chapter. The main objective of the ADA's is detection of zero-day worms and viruses broadcasting ARP requests to find vulnerable hosts. Besides, the approach will be very effective in preventing unwanted traffic, too.

4. Anomaly Detection by Stochastic Learning Automata

In this section the proposed method based on *Stochastic Learning Automata* (SLA) is described. A learning algorithm that constructs host-based learning models of normal ARP behavior from attack-free network ARP traffic is presented. Behavior that deviates from the learned normal model signals possible novel attacks.

4.1 Formal Description of SLA

An *automaton* is a machine or control mechanism designed to automatically follow a predetermined sequence of operations. The *stochastic* emphasizes the adaptive nature of the automaton. This adaptation is the result of *learning* process.

Formally, the automaton can be represented by quintuple $\{\Phi, \alpha, \beta, F(\bullet,\bullet), H(\bullet,\bullet)\}$ [K. S. Narendra & M. A. L. Thathachar, 1989], where :

- Φ is a set of internal *state*s. At any instant n, the state $\phi(n)$ is an element of the finite set Φ which is as follow:

$$\Phi = \{\phi_i \mid \forall i, 1 \le i \le s\} \quad (1)$$

- α is a set of actions (or outputs of the automaton). The *output* or *action* of an automaton at the instant n, denoted by $\alpha(n)$, is an element of the finite set α. Description of α is as below:

$$\alpha = \{\alpha_i \mid \forall i, 1 \le i \le r\} \quad (2)$$

- β is a set of responses (or inputs from the environment). The input from the environment $\beta(n)$ is an element of the set β which could be either a finite set or an infinite set, such as an interval on the real line:

$$\beta = \{\beta_i \mid \forall i,\ 1 \le i \le m\} \text{ or } \beta = \{(a,b)\} \tag{3}$$

- $F(\bullet,\bullet)$: $\Phi \times \beta \rightarrow \Phi$ is a function that maps the current state and input into the next state. F can be deterministic or stochastic:

$$\phi(n+1) = F[\phi(n), \beta(n)] \tag{4}$$

- $H(\bullet,\bullet)$: $\Phi \times \beta \rightarrow \alpha$ is a function that maps the current state and input into the current output. If the current output depends on only the current state, the automaton is referred to as state-output automaton. In this case, the function $H(\bullet,\bullet)$ is replaced by an output function $G(\bullet)$: $\Phi \rightarrow \alpha$, which can be either deterministic or stochastic:

$$\alpha(n) = G[\phi(n)] \tag{5}$$

The automaton applied for our application is of type of the later case.

4.2 General Reinforcement Scheme

In order to describe the reinforcement scheme, $p(n)$ is defined as a vector of action probabilities :

$$P_i(n) = P(\ \alpha(n) = \ \alpha_i\ )\ ,\ 1 < i < r \tag{6}$$

Updating action probabilities can be represented as follow:

$$P(n+1) = T[p(n),\ \alpha(n),\ \beta(n)] \tag{7}$$

where T is a mapping. This formula says the next action probability $p(n+1)$ is updated based on the current probability $p(n)$, the input from the environment and the resulting action.
The general scheme for updating action probabilities for an r-action automaton in an environment with $\beta = \{0,\ 1\}$ is as follow:
if $\alpha(n) = \alpha_i$, when $\beta = 0$:

$$p_j(n+1) = p_j(n) - g_j(p(n)), \quad \forall j,\ j \neq i \tag{8.a.1}$$

$$p_i(n+1) = p_i(n) + \sum_{\substack{k=1 \\ k \neq i}}^{r} g_k(p(n)) \tag{8.a.2}$$

and when $\beta = 1$:

$$P_j(n+1) = p_j(n) + h_j(p(n)), \quad \forall j\ ,\ j \neq i \tag{8.b.1}$$

$$p_i(n+1) = p_i(n) - \sum_{\substack{k=1 \\ k \neq i}}^{r} h_k(p(n)) \tag{8.b.2}$$

Where g_k and h_k, $k = 1, 2, \ldots, r$ are continuous, nonnegative functions with the following assumptions :

$$0 < g_k(p(n)) < p_k(n) \tag{9.a}$$

$$0 < \sum_{\substack{k=1 \\ k \neq i}}^{r} p_k(n) + h_k(p(n)) < 1, \;\; 1 < i < r, \;\; 0 < p_k < 1 \tag{9.b}$$

4.3 Why Using SLA

Nowadays in *Intrusion Detection* researches, efforts are mainly focused on misuse detection direction, since it is strait forward and easy to implement. But it has some inherent disadvantages. It is difficult to gather required information on known attack (content of TCP packets must be checked while maybe not enough). The most severe disadvantage is that it possibly can not detect attempts to new and unforeseen vulnerabilities.

These disadvantages make *Anomaly Detection* approaches a vibrant research area. Here we will make some effort to do host-based anomaly detection.

In order to model normal user's behavior, we believe that a good model should be able to give a reasonable explanation of the real system. Here SLA is used to satisfy this condition. Our reasons are as follow:

1. A computer user of a system should have some kind of routine behavior, especially for long-term computer users. This is what anomaly detection is based on.
2. Each computer user should be in some kind of state, when using computer. This state corresponds to what he currently mainly wants to do. For example, at one time, the user wants to browse web sites for shopping, at another time, he wants to make programming, etc. In each state, the user will mainly do some correspondent actions which are different with other states. So from statistic aspect, the distribution of every kind of connections or commands in each state will be different from other states.
3. Transition from one state to another can be treated roughly as state transition process of *Finite State Machine*. For the *Steady State Duration* time, we treat it as Gaussian (Normal) distribution, since human doing a task is not without remembering, so exponential distribution can not be used. On state transition decision, because human usually make decision on which task he will do next based on the previous several tasks he has done, so we treat the transition probability with conditional transition.

So from above three aspects, we believe SLA can be used for modeling of computer user's behavior in an understandable and accurate way.

4.4 Modeling Normal Behavior of ARP Traffic with SLA

For each node of network one automaton is learned from attack-free network ARP traffic.

In this approach, there is one state corresponding to each node in the network. So that, the set of internal states for each node learning automaton is defined as follow:

$$\Phi = \{IP_i \mid 0 \leq i \leq s\} \tag{10}$$

Where IP_i is the IP address of node i and s is the number of existing nodes in the network.

The set of actions (or outputs of the automaton) is a set of triples as follow:

$$\alpha = \{(IP_i,\ \varepsilon_i,\ \sigma_i^2)\ |\ \forall i,\ 1 \le i \le r\} \tag{11}$$

Where IP_i is the state identity and ε_i and σ_i^2 are the Average and Variance of steady state duration, respectively which are defined as below:

$$\varepsilon_i = \frac{\sum_{j=1}^{n_i} t_{ij}}{n_i} \tag{12}$$

$$\sigma_i^2 = E[t_i^2] - E[t_i]^2 \tag{13}$$

Where t_{ij} is the elapsed time after j^{th} ARP request with destination IP address corresponding to i until next ARP request with source IP address i issues. n_i is the number of occurrence of the i (i.e., the number of ARP requests issued with destination IP address corresponding to state i).

$E[t_i]$ and $E[t_i^2]$ in the second expression are expected values (mean) of the random variables t_i (steady state duration of state i) and t_i^2, respectively.

There is one action (output) for each state of the automaton, so we have ($r = s$).

The environment (network in our model) interacts with this automaton by introducing ARP requests to it. β, the set of responses (or inputs from environment) is defined as follow:

$$\beta = \{req_i\ |\ 0 \le i \le m\} \tag{14}$$

Where req_i is ARP request with destination IP address i. As stated earlier, the normal model is learned for each node x in the network. Therefore, the source IP address of all of the members of set β is same as the node whose normal model is under learning. It is obvious that m in this definition is equal to the number of nodes. We have ($m = s$).

Each ARP request causes a transition from one state (the state corresponding to destination IP address of the previous ARP request) to another state (the state corresponding to the ARP request destination IP address). The formal description of transition function (F) is as stated below:

$$IP_{n+1} = F\ (IP_n,\ req_{n+1}) \tag{15}$$

The transition function F of the automaton is deterministic and the result of this function is uniquely specified for each state. For each special state X and req_Y issued from the environment, the automaton changes its state from X to Y and this is deterministic.

The current output of the model is dependant on only current state, so the automaton is state-output. The formal description of output function is as below:

$$G(IP_n) = (IP_n,\ \varepsilon_n,\ \sigma_n^2) \tag{16}$$

This function is stochastic and nondeterministic, because the output set α is updated whenever the environment interacts with the automaton (whenever an ARP request issues). The elements of the set G are denoted by g_{ij}. The value of this element represents the probability that the action performed by the automaton is $(IP_j, \varepsilon_j, \sigma_j^2)$ given the automaton is in state IP_i :

$$G_{ij} = P[\alpha(n) = (IP_j, \varepsilon_j, \sigma_j^2) \mid \Phi(n) = IP_i], \quad 1 < i, j < s \tag{17}$$

In short the automaton takes an input from the environment and produces an action based on this.

The automaton is a variable-structure one. Although, transition function F is deterministic and does not change over time, but the output function is stochastic and its value changes over time.

4.5 The Reinforcement Scheme of the Proposed Model

The reinforcement scheme is the basis of learning process for learning automata. The formal definition of reinforcement scheme can be obtained by substitution of functions in Equation (8) as follow:

If $\Phi(n) = IP_n$ and the environment issues req_n then award function will be as:

$$g_k(p(n)) = \frac{a}{n+1} \tag{18}$$

If $\Phi(n) = IP_n$ and the environment issues req_m, $m \neq n$ then penalty function will be as:

$$h_k(p(n)) = \frac{b}{n+1} \tag{19}$$

Therefore, the formal definition of the reinforcement scheme is given as:

if $\alpha(n) = IP_n$, when environment response is req_n:

$$p_j(n+1) = p_j(n) - \frac{a}{n+1}, \quad \forall j, j \neq i \tag{20.a.1}$$

$$p_i(n+1) = p_i(n) + \frac{(r-1).a}{n+1} \tag{20.a.2}$$

and when environment response is req_m, $m \neq n$:

$$p_j(n+1) = p_j(n) + \frac{b}{n+1}, \quad \forall j, j \neq i \tag{20.b.1}$$

$$p_i(n+1) = p_i(n) - \frac{(r-1) \times b}{n+1} \tag{20.b.2}$$

4.6 Anomaly Detection

The purposed algorithm constructs a learning model of normal ARP traffic for each existing host in the network. The model resulted from this learning process is named as *normal model*. Discussion of producing normal ARP traffic will come, latter. Online network traffic is compared by the normal model in a process referred to as "*matching process*". Any deviation from the normal model is an indication of anomaly which is quantified as *Anomaly Score* (AS) parameter. The algorithm makes decisions on normality or abnormality of each node by means of the calculated AS parameter in matching process.

An important parameter in anomaly detection is an accurate threshold value. An indication of normality is used for this purpose, referred to as *Normal Score* (NS). This parameter is described latter.

- *Anomaly Score* (AS)

We apply AS as an anomaly indicator of a node ARP traffic and obtain it from weighted summation of *Partial Anomaly Scores* (*PAS*'es) as the following equation denotes:

$$AS = K_S A_S + \sum_{n=2}^{N} \frac{K_j^n A_j^n}{P_{ij}} \tag{21}$$

Where N in the above equation is the number of previous environment responses (ARP requests issued in matching process) affecting on the AS value and decision making on normality or abnormality of the corresponding node, j is the state in the learning model which the node will be in after n^{th} ARP request and A_j^n is the *PAS*, explained latter.

A_S is the *PAS* corresponding to the state which the node will be inside after the first ARP request ((N-1)th previous ARP request) and can be stated formally as follow:

$$A_S = A_i^1, \quad 1 \le i \le r \tag{22}$$

K_j^n is coefficient of the participating term in weighted summation of AS and is dependent on state j in learning model, such that occurrence of a state with low probability has more effect on AS than occurrence of a state with high probability. So the purposed value for this parameter is simply the inverted state probability:

$$K_j^n = P_j(n)^{-1} \tag{23}$$

This justification can be described for transition probability residing in denominator. $P_{ij}(n)$ is conditional probability of transition from state i to state j caused by n^{th} ARP request (environment response req_j^n), given the sequence of observed transitions in matching process. It can be formally described as follow:

$$P_{ij}(n) = P(T_{ij} \,|\, T_{I_1 I_2} T_{I_2 I_3} \ldots T_{I_{n-2} I_{n-1}}) \tag{24}$$

Where, T_{ij} is transition from state i to state j, $T_{I_1 I_2} T_{I_2 I_3} \ldots T_{I_{n-2} I_{n-1}}$ is sequence of transitions in matching process, I_x indexes correspond to states where the node will be in after n^{th} ARP request in matching process (after $req^n_{I_n}$), as I_1 and I_{n-1} indexes correspond to states S and i, respectively. This conditional probability is calculated in learning process as follow:

$$P_{ij}(n) = \frac{P(T_{I_1I_2}T_{I_2I_3}...T_{I_{n-2}i}T_{ij})}{P(T_{I_1I_2}T_{I_2I_3}...T_{I_{n-2}i})} \tag{25}$$

***- Partial Anomaly Score* (*PAS*)**

Defined as deviation from average steady state duration in a state:

$$A^n_j = \begin{cases} \frac{(\varepsilon^n_j - t^n_j)^2}{(\sigma^n_j)^2}, & \text{if } \mathrm{t}^{\mathrm{n}}_{\mathrm{j}} < \varepsilon^n_j \\ 0, & \text{if } \mathrm{t}^{\mathrm{n}}_{\mathrm{j}} \geq \varepsilon^n_j \end{cases} \tag{26}$$

Where t^n_j is the time interval between n^{th} and $(n+1)^{\text{th}}$ ARP requests in matching process, such that the node will be in state j after n^{th} ARP request.

Steady state duration values, greater than average value (ε_j^{n}) for each state, indicate normal behaviors, so its effect does not participate in AS.

There are some states in learning model which environment doesn't interact with them (there is not any req_i for such states labeled with i). For such states we have:

$$\alpha_i = (IP_i, \varepsilon_i, \sigma_i^2) = (IP_i, 0, 0) \tag{27.a}$$

$$P_i(n) = 0 \tag{27.b}$$

A similar problem exists about sequences of transitions with these conditions. Minimum state probability among probabilities of all of the existent states is used for probability of such states and minimum probability among sequences of transitions corresponding to the node for the probability of such sequences of transitions, as formally stated below:

$$P_i(n) = MIN\ \{P_k(n)\ |\ 1 \leq k \leq r\} \tag{28.a}$$

$$P_{ij}(n) = MIN\ \{P_{ik}(n)\ |\ 1 \leq k \leq r\} \tag{28.b}$$

Also, we considered statistical parameters of each state i satisfying conditions stated in equation (27) (parameters ε_i, and σ_i^2), as follow:

$$\varepsilon^n_i = MAX\ \{\varepsilon^n_k\ |\ 1 \leq k \leq r\} \tag{29.a}$$

$$\sigma^n_i = MIN\ \{\sigma^n_k\ |\ 1 \leq k \leq r\} \tag{29.b}$$

It means taking the worst case value for these parameters.

***- Normal Score* (*NS*)**

NS, as hinted above, is an indicator of normality degree and is a function of partial *NS*'es (*PNS*). We define PNS_i as normality score at i^{th} time interval in learning process. *NS* is calculated as the following equation states:

$$NS = MAX \{PNS_i\} \tag{30}$$

For calculating PNS_i, we use the same method as used for calculation of AS, but in this case for normal ARP traffic. We get this normal traffic from purified traffic in different time intervals and calculate *PNS* for each interval. It is obvious that to obtain a right value for *NS* from *PNS*'s, these time intervals should be of the same length.

- Threshold calculations

An estimation of ARP traffic normality is required for a right threshold values. The *NS* value gives this estimation to the hand. It is an indication of maximum value of AS without being detected as abnormal. AS_i values calculated in matching process, satisfying the inequality $NS_i \leq Th_i < AS_i$ for threshold value of Th_i of node i, are detected as abnormal. Making decisions on *Th* values affect on *False Negative* and *False Positive* and accuracy of algorithm. There are various ways to this problem. One most simple and feasible way is to get threshold k times of *NS*. For example, the chosen value for *k* is 1.2 in [Kai Hwang, Hua Liu & Ying Chen, 2004].

5. Anomaly Detection by K-Means and ID3 Decision Trees

5.1 Anomaly Detection by K-Means Algorithm

The K-Means algorithm [R. Duda et al., 2000] groups *N* data points into *k* disjoint clusters, where *k* is a predefined parameter such that $k < N$. The steps in the K-Means clustering-based anomaly detection method are as follows:

1. Select *k* random instances from the training data subset as the centroids of the clusters $C_1, C_2, \ldots, C_k$.
2. For each training instance *X*:
    a. Compute the Euqlidean distance:

    $$D(C_i, X), i = 1...k.$$

    Find cluster C_q that is closest to *X*.

    b. Assign *X* to C_q. Update the centroid of C_q. (The centroid of a cluster is the arithmetic mean of the instances in the cluster.)
3. Repeat Step 2 until the centroids of clusters $C_1, C_2, \ldots, C_k$ stabilize in terms of mean-squared-error criterion. Finally, the algorithm aims at minimizing an *objective function* (here, squared error function). The objective function *J*:

    $$J = \sum_{j=1}^{k} \sum_{i=1}^{N} \left\| X_i^{(j)} - c_j \right\|^2 \tag{31}$$

    where the term:

    $$\left\| X_i^{(j)} - c_j \right\|^2$$

    is a chosen distance measure between a data point $X_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *N* data points from their respective cluster centers.

5.2 Anomaly Detection by ID3 Decision Trees

The ID3 decision tree learning algorithm [T. Mitchell, 1997] computes the *Information Gain G* on each attribute *A*, as:

$$G(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{32}$$

where *S* is the total input space and S_v is the subset of S for which attribute *A* has a value *v*. The *Entropy(S)* over *c* classes is given by:

$$Entropy\ (S) = \sum_{i=1}^{c} - p_i \log_2 (p_i) \tag{33}$$

where p_i represents the probability of class "*i*". The probability of class *i* is calculated as follow:

$$p_i = \frac{N_i}{\sum_{j=1}^{c} N_j} \tag{34}$$

where, N_x is the number of training instances in class *x*.

The attribute with the highest information gain, say B, is chosen as the root node of the tree. Next, a new decision tree is recursively constructed over each value of B using the training subspace *S-{S_B}*. A leaf-node or a decision-node is formed when all the instances within the available training subspace are from the same class. The algorithm constructs the ID3 decision tree with the normal purified traffic. Anomaly detection is performed using this tree by traversing the tree with features of test instance. If the traverse reaches a leaf node, the test instance will be detected as normal; else it will be detected as abnormal.

5.3 Anomaly Detection by Combined K-Means Clustering & ID3 Decision Trees

We are provided with a normal (purified) training data set X_i where each instance represents an n-dimensional vector. The approach has two phases: training and testing. During training phase, K-Means-based anomaly detection method is first applied to partition the training space into *k* disjoint clusters $C_1, C_2, \ldots, C_k$. Then ID3 decision tree is trained with instances in each K-Means cluster. The K-Means clustering method ensures that each training instance is associated with only one cluster. However, if there are any subgroups or overlaps within a cluster, the ID3 decision tree trained on that cluster refines the decision boundaries by partitioning the instances with a set of if-then rules over the feature space.

The combined application of the two algorithms overcomes some limitations of each algorithm when applied individually. For example, selection of a right value for parameter *k* in the K-Means clustering algorithm can affect on the overall accuracy of the algorithm. Considerably little values of *k*, compared to inherent number of natural subgroupings within the training data will lead to overlapping subgroups within clusters. This problem is compensated by ID3 decision tree constructed in each cluster.

The testing phase of the algorithm includes two phases: the candidate selection phase and candidate combination phase. In the first phase, AS from K-Means clustering and decisions from ID3 decision tree are extracted. In the second phase, the final AS is gotten from combined results of K-Means clustering and ID3 decision tree. The algorithms applied for candidate selection and candidate combination is explained below.

5.4 The Candidate Selection Phase

Steps of the candidate selection algorithm are as follow:

1. For each test instance Z_i, $1 \le i \le n$:
    a. Compute Euclidean distance:

    $$D(Z_i, r_j)\ , j = 1 < j < n,$$

    and find f clusters closest to Z_i.

    b. Compute K-Means AS and extract decisions of ID3 decision trees for f nearest candidate clusters.
2. Return Anomaly Score Matrix for Z_i.

The algorithm gets test instances Z_i, $1 \le i \le n$, and the parameter f as inputs and gives matrix AS$[f\times 2]$ for each test instance Z_i. f is a user defined parameter. If $DT_1, DT_2, \ldots, DT_k$ be the ID3 decision tress on clusters $C_1, C_2, \ldots, C_k$ formed by applying the K-Means method on the training instances, and $r_1, r_2, \ldots, r_k$ be the centroids of clusters $C_1, C_2, \ldots, C_k$, respectively, then given a test instance Z_i, the candidate selection procedure extracts AS'es from f candidate clusters $G_1, G_2, \ldots, G_k$. The selected f candidate clusters are f clusters in $C_1, C_2, \ldots, C_k$ that are nearest to Z_i in terms of Euclidean distance between Z_i and the cluster centroids.

Let $l_1, l_2, \ldots, l_f$ be the centroids of candidate clusters $G_1, G_2, \ldots, G_f$ and the Euclidean distances between the test vector Z_i and the f candidate clusters is as follow:

$$D(Z_i, l_j) = d_j, \quad 1 \le j \le f \tag{35}$$

the AS for each of the f candidate clusters is calculated by K-Means clustering as follow:

$$AS_j = P(C_j) \times [1 - \frac{d_j}{\sum_{a=1}^{k} D(Z_i, r_l)}], \;\; 1 \le i \le n, \;\; 1 \le j \le f \tag{36}$$

$P(C_j)$ in the above equation is probability of cluster C_j and is calculated as the following equation indicates:

$$P(C_j) = \frac{N_{C_j}}{\sum_{i=1}^{k} N_{C_i}} \tag{37}$$

where the nominator in the above equation is the number of training instances of cluster C_j and the denominator is the total number of training instances.
Term:

$$1-\frac{d_j}{\sum_{a=1}^{k} D(Z_i, r_l)}$$

in equation (36) is referred to as *Scaling Factor* (SF). It scales $P(C_j)$ by weighing it against the ratio of the Euclidean distance between the cluster j and Z_i, and the sum of Euclidean distances between Z_i and the clusters $C_1, C_2, \ldots, C_k$. The SF penalizes the probability of cluster C_j with its distance from the test vector Z_i, such that little distance D_j yields a high AS_j and voice versa. The decision from the ID3 decision trees associated with the f candidate clusters are either "0" or "1" representing normal or abnormal test instances, respectively. This output is depended on that the decision trees can be traversed reaching to a leaf node.
For each test instance Z_i, the candidate selection phase outputs a matrix AS$[f\times 2]$ with AS calculated by K-Means clustering algorithm and decision extracted from ID3 decision tree. The final AS resulted from combined application of the two algorithms is extracted by Candidate Combination. A detailed explanation of this algorithm follows.

5.5 The Candidate Combination Algorithm

Candidate Combination algorithm take as input the output of Candidate Selection algorithm, the AS matrix including the AS_j, $1 < j < f$, values of the K-means clustering algorithm and the decisions of ID3 decision trees over f candidate clusters. The algorithm then order the f candidate clusters $G_1, G_2, \ldots, G_k$ in AS matrix such that the distances $d_1, d_2, \ldots, d_f$ between Z and the candidate clusters $G_1, G_2, \ldots, G_f$, respectively, satisfy $d_1 < d_2 < \ldots < d_f$. The first AS_j value of the K-means clustering algorithm in the ordered AS matrix satisfying each of the below conditions, is selected as the final AS value of the combined K-Means clustering and ID3 decision tree algorithm. The conditions are:

$$AS_j \leq 0.5 \ \& \ DT_j = 0$$
$$AS_j > 0.5 \ \& \ DT_j = 1$$

Where DT_j = "0" or DT_j = "1", means ID3 decision tree of cluster j classifies the test instance as normal or abnormal, respectively. Finally, for each test instance Z_i, an AS value in continuous closed interval [0,1] is yielded from the combined application of the two algorithms. The Threshold Rule is used for classification of the test instance Z_i as an anomalous or normal instance. The threshold rule for classifying a test instance Z_i that belongs to cluster C_r is as follow:

$$\textit{Assign } Z \rightarrow 1 \textit{ if } AS > \tau,$$
$$\textit{Otherwise } Z \rightarrow 0.$$

where τ is a predefined threshold.

6. Evaluation Test Bed Network and Data Set

Our test bed network in this research work was a typical network with about 900 active devices. With exception of a few servers, all of hosts run different Microsoft Windows platforms like windows 98, 2K, XP and 2003 Server. The network is connected to Internet via a 2Mbps link and about 200 stations and 5 servers are connected to internet concurrently. Internet access was enabled for majority of hosts.

The captured data set contains approximately, eight weeks, three weeks of training and two weeks of test data. The main resources of abnormalities in our evaluation test bed network are malicious softwares, malfunctioning network devices, ill-advised applications, scanning tools and high level user's efforts for network discovery. To capture network traffic we used a computer connected to the core switch of the network. Capturing traffic and some statistical parameters from it, is performed in real-time interaction with our prototype, by setting the sniffing machine's NIC in sniffing mode. A sniffing tool in VC++ powered by WinPcap application programming interface has been developed for traffic capturing. The implementations have been performed by Matlab V.7.5.0.342.

Some anomaly criteria are defined and applied to the captured ARP traffic to generate normal training instances. These anomaly criteria are as follow:

- ARP rate: this criterion is defined as the overall number of ARP requests divided by the length of time over which these were observed.
- Burstiness: if we define the maximum instantaneous ARP request rate for a device to be the inverse of the shortest observed inter-request time between two consecutive requests from that devise, burstiness can be defined as the ratio of maximum request rate to the ARP rate. The burstiness characteristics of ARP traffic for our evaluation test bed network are illustrated in figure (1). This diagram shows that most devices in normal operation do not send ARP request broadcasts in bursts.

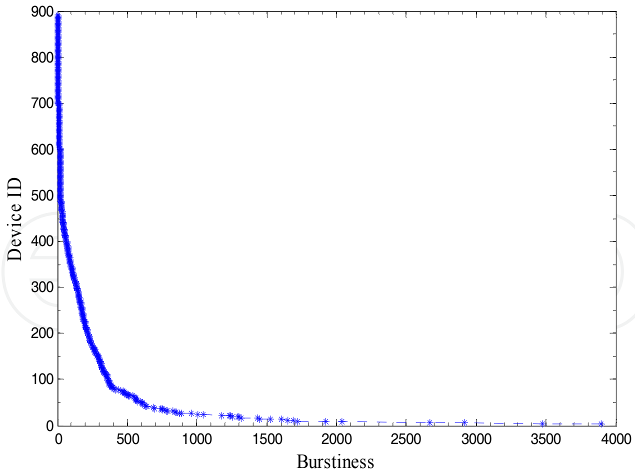


Fig. 1. Burstiness characteristics of ARP traffic for our test network.

- Sequential scans: sequential scan is defined as ARP requests with sequential destination IP addresses. ARP requests in normal conditions have not sequential destination IP addresses.
- Dark space: is defined as ARP requests with destination IP addresses not included in address space of network.

- Repetitive Requests: this criterion is defined as ARP Requests within time intervals smaller than expiration time of corresponding entries in ARP tables. ARP tables maintained by each host or network device are updated when an ARP request is issued. This caching mechanism prevents repetitive ARP requests.

7. Evaluation & experimental results

Our evaluation is based on the following criteria:

- *Sensitivity*: probability that a test result will be positive when there is anomaly (*True Positive* or TP).
- *Specificity*: probability that a test result will be negative when the there is not anomaly (*True Negative* or TN).
- *Negative likelihood ratio*: ratio between the probability of a negative test result given the *presence* of the anomaly and the probability of a negative test result given the *absence* of the anomaly, i.e. Negative likelihood ratio = False negative rate / True negative rate = (1-Sensitivity) / Specificity.
- *Positive predictive value*: probability that the anomaly is present when the test is positive.
- *Negative predictive value*: probability that the anomaly is not present when the test is negative.

  ARP traffic has been applied to detect network abnormalities in the approach, as stated before. In K-Means clustering, ID3 Decision trees, and the combinatorial approach of these two algorithms we are provided with training and test data set X_i , $1 \le i \le N$, where X_i represents a 9-dimentional vector as follow:

$$X_i=(S_{i1}, D_{i1}, \Delta T_{i1}, S_{i2}, D_{i2}, \Delta T_{i2}, S_{i3}, D_{i3}, \Delta T_{i3}) \qquad (38)$$

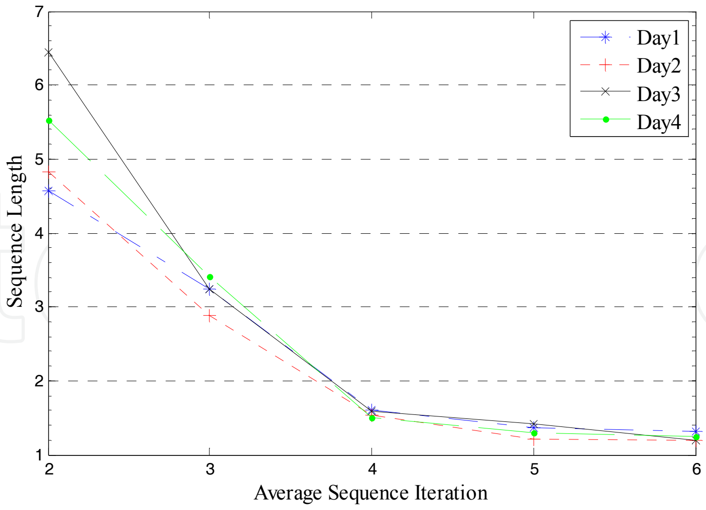


Fig. 2. Iteration of sequences with different length.

S_{ix}'es and D_{ix}'es in the data instance X_i are source and destination IP addresses of ARP requests. We have used three successive ARP request characteristics in each data instance X_i. ΔT_{ix} is the discretely quantized time interval between two successive ARP requests. The main reason for using characteristics of multiple ARP requests in each data instance X_i, is that user activities include some sequential activities which are depended on the state the user is in and what he mainly wants to do in that state. So, individual ARP requests can not be applied for detecting abnormalities. Figure (2) presents average iteration of ARP requests sequences with different length within four different days. As it is obvious from this figure, there is an egregious difference between iteration of sequences of length 3, 4 such that iteration of sequences of length 4 is as close to 1 as desired. So, we have used three successive ARP requests in each data instance X_i.

The experimental results within five successive weeks are represented in table (1). The experiments are based on five evaluation measures, described above. The Sensitivity, Specificity, Negative Likelihood Ratio, Positive Predictive Value, and Negative Predictive Value characteristics of K-Means clustering, ID3 Decision tree, SLA-based and the combinatorial K-Means+ID3 is illustrated in figures (3) to (7). These figures show that:

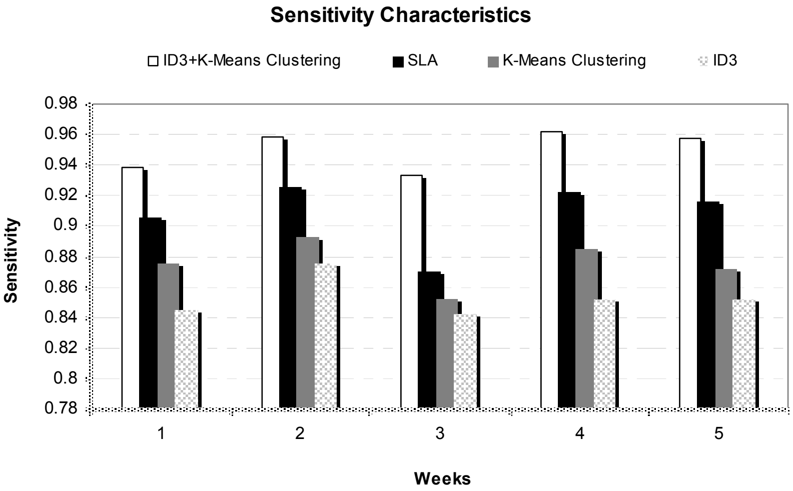


Fig. 3. Comparison of Sensitivity characteristics.

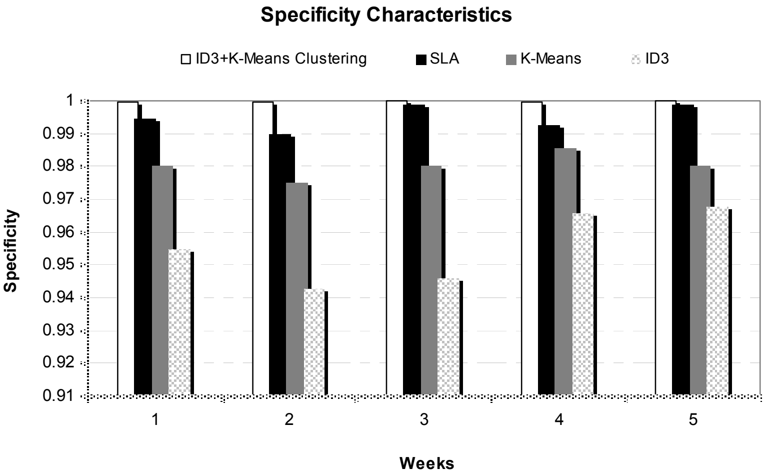


Fig. 4. Comparison of specificity characteristics.

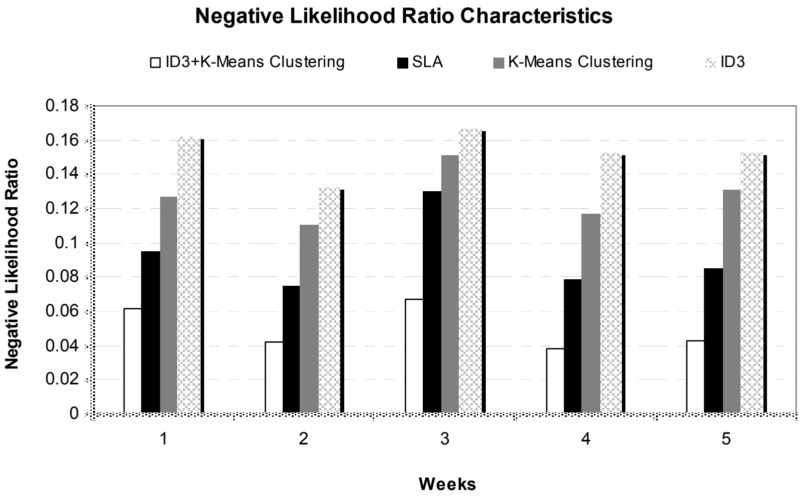


Fig. 5. Comparison of negative likelihood ratio characteristics.

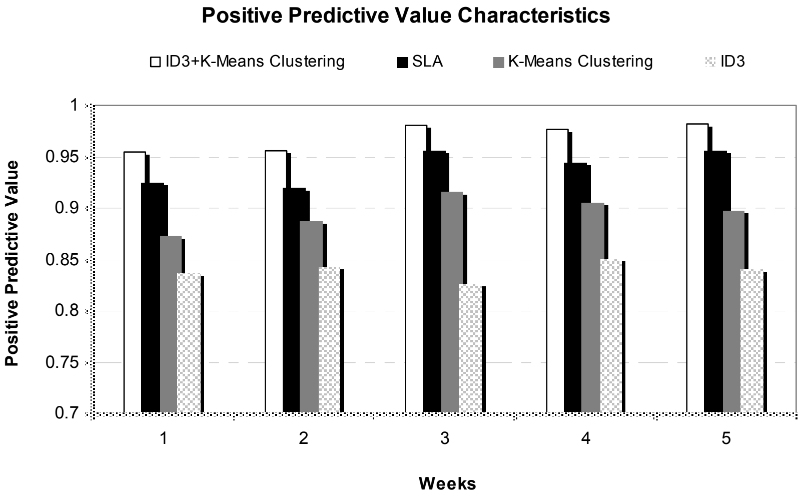


Fig. 6. Comparison of positive predictive value characteristics.

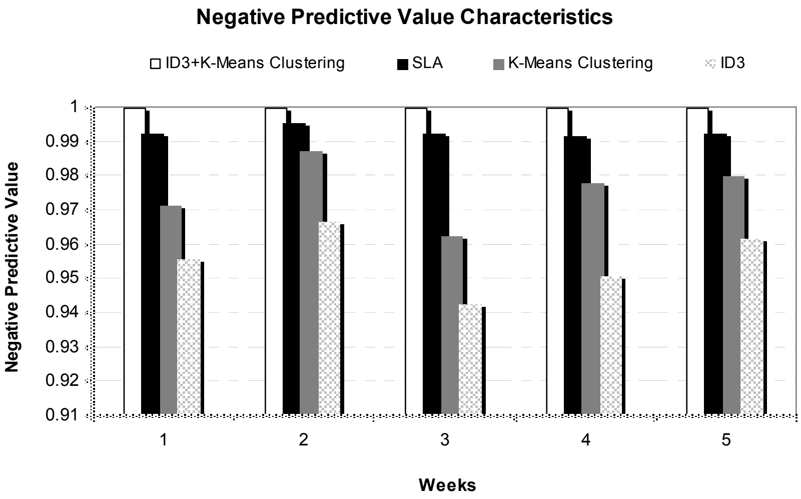


Fig. 7. Comparison of negative predictive value characteristics.

Week	1	2	3	4	5
Total Number of Test Instances	679100	856200	987000	598400	832700
The Number of Abnormality Instances	4500	10900	3400	5900	4200
Sensitivity	0.938326	0.958148	0.933155	0.961538	0.957303
Specificity	0.999704	0.99942	0.999929	0.999747	0.999903
Negative Likelihood Ratio	0.061692	0.041876	0.06685	0.038471	0.042701
Positive Predictive Value	0.955157	0.956444	0.980337	0.976563	0.981567
Negative Predictive Value	0.999585	0.999444	0.999746	0.999578	0.999771

Table 1. Evaluation results of the K-means+ID3 in five weeks.

1) K-Means+ID3 method has better performance than the other methods in terms of all defined measures.
2) The performance of the SLA-based approach is in-between the combinatorial approach and each of individual K-Means and ID3.
3) Individual K-Means has better performance than individual ID3.

Malicious softwares by issuing a large number of ARP packets in little time intervals had large effects on traffic abnormalities. They assigned high percentage of triggered alarms to themselves. Bad-configured applications were the main origins of abnormality after malicious softwares. Curious users' activities, malfunctioning or bad configured network devices, and etc. affect on network traffic abnormalities but had a lower portion in it.

8. Conclusion

This chapter presented some anomaly detection approaches for classification of anomalous and normal activities in computer network ARP traffic. The proposed approaches use some well-known machine learning methods: the SLA, K-Means clustering and the ID3 decision tree learning approaches. As described, in SLA-based approach a learning algorithm has been used for modeling of normal ARP traffic behavior. Making decisions on abnormal behavior of each device in the network is based on comparison of online behavior of each host by its normal model. In the combinatorial approach based on K-means and ID3 decision trees, the K-Means method was first applied to partition the training instances into *k* disjoint clusters. The ID3 decision tree built on each cluster learns the subgroups within the cluster and partitions the decision space into finer classification regions; thereby improving the overall classification performance. This combinatorial method was compared with the individual K-Means and ID3 methods and the other proposed approaches based on SLA in terms of the overall classification performance defined over five different performance measures. Results on real evaluation test bed network data sets show that: the proposed method outperforms the individual K-Means and the ID3 compared to the other approaches. The performance of SLA is in-between the proposed combinatorial K-Means+ID3 and individual K-Means and ID3, in terms of all the five performance measures over the real network ARP traffic data set.

Further research should be carried out to evaluate the performance of the proposed approaches with other combinatorial approaches which can be developed by different clustering approaches.

Acknowledgements

This work was partially supported by the Payam Nour University and Iran Information Technology Research Center.

9. References

A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection", *Proceedings of SIAM Int'l Conference Data Mining*, May 2003.

A. Ray, "Symbolic Dynamic Analysis of Complex Systems for Anomaly Detection", *Signal Processing*, vol. 84, no. 7, pp. 1115- 1130, 2004.

A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis, "Soft Combination of Neural Classifiers: A Comparative Study", *Pattern Recognition Letters*, vol. 20, pp. 429-444, 1999.

C. Kruegel and G. Vigna, "Anomaly Detection of Web-Based Attacks", *Proceedings of ACM Conference Computer and Communication Security*, Oct. 2003.

D. Ármannsson, G. Hjálmtýsson, P. D. Smith, L.Mathy; "Controlling the Effects of Anomalous ARP Behaviour on Ethernet Networks", *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 2005, pp. 50–60.

D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, "Anomalous System Call Detection," *ACM Trans. Information and System Security*, vol. 9, no. 1, pp. 61-93, Feb. 2006.

D. Whyte, E. Kranakis, P. Van Oorschot, "ARP-Based Detection of Scanning Worms within an Enterprise Network", *Proceedings of Annual Computer Security Applications Conference* (*ACSAC 2005*), Tucson, AZ, Dec. 5-9, 2005.

D. Y. Yeung and C. Chow, "Parzen-Window Network Intrusion Detectors", *Proceedings of 16th Int'l Conf. Pattern Recognition*, vol. 4, pp. 385- 388, Aug. 2002.

G. Maselli, L.Deri, "Design and Implementation of an Anomaly Detection System: an Empirical Approach", *Proceedings of Terena TNC 2003*, May 2003, Zagreb, Croatia.

G. Qu, S. Hariri, and M. Yousif, "A New Dependency and Correlation Analysis for Features", *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 9, pp. 1199-1207, Sept. 2005.

H. S. Javitz and A. Valdes, "The SRI IDES Statistical Anomaly Detector", *Proceedings of IEEE Symp. Security and Privacy*, pp. 316-326, May 1991.

I. Levin, "KDD-99 Classifier Learning Contest: LLSoft"s Results Overview", *SIGKDD Explorations*, vol. 1, pp. 67-75, Jan. 2000.

J. Gomez and D.D. Gupta, "Evolving Fuzzy Classifiers for Intrusion Detection", *Proceedings of 2002 IEEE Workshop Information Assurance*, June 2001.

J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, "On Combining Classifiers", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226-239, Mar. 1998.

K. Hwang, H. Liu, and Y. Chen, "Protecting Network-Centric Systems with Joint Anomaly and Intrusion Detection over Internet Episodes", *IEEE IPDPS- 2005*, Oct.8, 2004.

K. S. Narendra, M. A. L. Thathachar, Learning Automata: an introduction, Prentice-Hall, 1989.

Kai Hwang, Hua Liu, Ying Chen , "Cooperative Anomaly and Intrusion Detection for Alert Correlation in Networked Computing Systems", *IEEE Trans. Dependable and Secure Computing*, November 24, 2004.

L. I. Kuncheva, “Switching between Selection and Fusion in Combining Classifiers: An Experiment”, *IEEE Trans. Systems, Man, and Cybernetics*, vol. 32, no. 2, pp. 146-156, Apr. 2002.

M. Thottan and C. Ji, “Anomaly Detection in IP Networks”, *IEEE Trans. Signal Processing,* vol. 51, no. 8, pp. 2191-2204, 2003.

N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, “Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection”, *IEEE Trans*. Computers, vol. 51, no. 7, pp. 810-820, 2002.

N. Ye, Y. Zhang, and C.M. Borror, “Robustness of the Markov-Chain Model for Cyber-Attack Detection”, *IEEE Trans. Reliability*, vol. 53, no. 1, pp. 116-123, 2004.

R. Agarwal and M.V. Joshi, “PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection)”, *Technical Report DSTO-GD-0286*, Dept. of Computer Science, Univ. of Minnesota, 2000.

R. Duda, P. Hart, and D. Stork, *Pattern Classification*, second ed. Wiley Publishers, Oct. 2000.

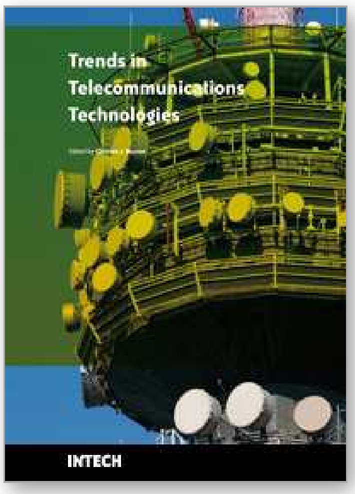
S. Kumar and E.H. Spafford, “A Pattern Matching Model for Misuse Intrusion Detection”, *Proceedings of 17th Nat’l Computer Security Conf.*, pp. 11-21, Oct. 1994.

S. T. Sarasamma, Q. A. Zhu, and J. Huff, “Hierarchical Kohonen Net for Anomaly Detection in Network Security”, *IEEE Trans. Systems, Man, and Cybernetics*-Part B, vol. 35, no. 2, Apr. 2005.

Shekhar R. Gaddam, Vir V. Phoha, Kiran S. Balagani, “K-Means+ID3: A Novel Method for Supervised Anomaly Detection by Cascading K-Means Clustering and ID3 Decision Tree Learnimg Methods”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 3, March 2007.

T. Mitchell, *Machine Learning*. McGraw-Hill, 1997**.**

Z. Zhang, J. Li, C.N. Manikopoulos, J. Jorgenson, and J. Ucles, “HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification”, *Proceedings of 2001 IEEE Workshop Information Assurance*, pp. 85-90, June 2001.



Trends in Telecommunications Technologies
Edited by Christos J Bouras

ISBN 978-953-307-072-8
Hard cover, 768 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

The main focus of the book is the advances in telecommunications modeling, policy, and technology. In particular, several chapters of the book deal with low-level network layers and present issues in optical communication technology and optical networks, including the deployment of optical hardware devices and the design of optical network architecture. Wireless networking is also covered, with a focus on WiFi and WiMAX technologies. The book also contains chapters that deal with transport issues, and namely protocols and policies for efficient and guaranteed transmission characteristics while transferring demanding data applications such as video. Finally, the book includes chapters that focus on the delivery of applications through common telecommunication channels such as the earth atmosphere. This book is useful for researchers working in the telecommunications field, in order to read a compact gathering of some of the latest efforts in related areas. It is also useful for educators that wish to get an up-to-date glimpse of telecommunications research and present it in an easily understandable and concise way. It is finally suitable for the engineers and other interested people that would benefit from an overview of ideas, experiments, algorithms and techniques that are presented throughout the book.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Yasser Yasami and Saadat Pourmozaffari (2010). New Trends in Network Anomaly Detection, Trends in Telecommunications Technologies, Christos J Bouras (Ed.), ISBN: 978-953-307-072-8, InTech, Available from: http://www.intechopen.com/books/trends-in-telecommunications-technologies/new-trends-in-network-anomaly-detection

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821