System Approach to Intrusion Detection Using Hidden Markov Model

Rahul Khanna
Intel Corporation, 2111 NE 25th Ave., Hillsboro, OR 97124, USA
rahul.khanna@intel.com

Huaping Liu
Oregon State University, School of EECS, Corvallis, OR 97331 USA
hliu@eecs.oregonstate.edu

ABSTRACT

In an era of cooperating ad hoc networks and pervasive wireless connectivity, we are becoming more vulnerable to malicious attacks. Many of these attacks are silent in nature and cannot be detected by the conventional intrusion detection system (IDS) methods such as traffic monitoring, port scanning, or protocol violations. These sophisticated attacks operate under the threshold boundaries during an intrusion attempt and can only be identified by profiling the complete system activity in relation to a normal behavior. In this paper we discuss a hidden Markov model (HMM) strategy for intrusion detection using a multivariate Gaussian model for observations that are then used to predict an attack that exists in a form of a hidden state. This model is comprised of a self-organizing network for event clustering, an observation classifier, a drift detector, a profile estimator, a Gaussian mixture model (GMM) accelerator, and an HMM engine. We use this method to predict the intrusion states based on observation deviation from normal profiles or by fitting it into an appropriate attack profile.

Categories and Subject Descriptors

G.3 [**Probability and statistics**]: Multivariate statistics

General Terms

Security Algorithms

Keywords

IDS, Hidden Markov Models, wireless ad hoc networks.

1. INTRODUCTION

Intrusion detection system (IDS) protects the data integrity and manages the system availability during intrusion. In a mobile ad hoc network (MANET) it must deal with challenges related to fully-mobile, self-configuring, multi-hop wireless networks with varying resources and limited bandwidth. Routing protocols like dynamic source routing (DSR), destination sequenced distance vector (DSDV), ad hoc on-demand distance vector (AODV), and secure ad hoc on-demand distance vector (SAODV), etc., collaborate assuming a trustworthy communication without a standard authentication and authorization mechanism thus exposing weak nodes. Existing IDS schemes include adaptive intrusion system (AID), graph based intrusion detection system (GrIDS), network anomaly detection and intrusion reporting (NADIR), and network intrusion detector (NID).

Unlike wired networks, ad hoc nodes coordinate between member nodes to allow exclusive use of the communication channel. A malicious node can exploit this distributed and complex decision making property of cooperating nodes to launch an attack or hijack the node [1]. This inherent vulnerability can disable the whole network cluster and further compromise the security by impersonating, message contamination, hijacking, passive listening, or acting as a malicious router. An IDS mechanism should be able to detect intrusion by monitoring unusual activities in the system by comparing it to a user's profile and evolving trends. While threshold based mechanisms may not be sufficient to prevent malicious attacks if the attacker operates below the threshold, it can be modified to monitor trends in the related system components to predict an attack. This is similar to a hidden Markov model (HMM), where the hidden state (attack) can be predicted from relevant observations (changes in system parameters, fault frequency, etc.). We define the concept of profile that characterizes the behavior of a given subject with respect to a given object, thereby serving as a signature or description of normal activity for its respective subject(s) and object(s). Observed behavior is characterized in terms of a statistical metric and model. A metric is a random variable representing a quantitative measure accumulated over a period. Observations obtained from the audit records when used together with a statistical model analyzes any deviation from a standard profile and triggers a possible intrusion state.

This paper proposes an HMM based approach that correlates the system observations (usage and activity profile) and state transitions to predict the most probable *intrusion state sequence*. An HMM is a stochastic models of discrete events and a variation of the Markov chain. Like a conventional Markov chain, an HMM consists of a set of discrete states and a matrix $\boldsymbol{A} = \{a_{ij}\}$ of *state transition probabilities*. In addition, every state has a vector of *observed symbol probabilities*, $\boldsymbol{B} = b_j(v)$ that corresponds to the probability that the system will produce a symbol of type v when it is in state j. The states of the HMM can only be inferred from the observed symbols, hence called "hidden".

2. RELATED WORK AND MOTIVATION

Intrusion detection could be be considered equivalent to

IWCMC'06, July 3–6, 2006, Vancouver, British Columbia, Canada.

an immune system that identifies and eliminates any anomalies. This requires understanding the specifications of the normal processes so that any anomalies can be identified. The identifiers should be distributed over the system with identifiable and adaptable relationship. We therefore need a model that, in each state, has a probability of producing observable system outputs and a separate probability indicating the next states. Ko *et al.* suggested a policy specification language to test the behavior of the prevliged programs [2] where any deviation from normal was tagged as a misuse. Wagner *et al.* proposed a statistical approach [3] which monitors the system call trace of a programs execution for compliance to the precomputed model. Ye *et al.* used system call frequencies and system call ordering from audit data to predict an intrusion [4]. Manganaris *et al.* proposed an alert based policy mechanism [5] that associates an alert with multiple events frequently occurring together.

HMM correlates observations with hidden states that factor in the system design where observation points are optimized using an acceptable set of system-wide intrusion checkpoints (IC) while hidden states are created using explicit knowledge of probabilistic relationships with these observations. These relationships, which are also called profiles, are hardened and evolved with the constant usage of the multiple and independent systems. If observation points can be standardized, then the problem of intrusion predictability can be reduced to profiling the existing and new hidden states to standard observations. For modeling a large number of temporal sequences, HMM can act as an excellent alternative, because it has been widely used for pattern matching in speech recognition [6], image identification [7], and Internet attacks [8]. If we consider an attack to be a pattern of an observed sequence, HMM should be appropriate to map those patterns to one of many attack states.

3. MODELING SCHEME

HMM modeling schemes consist of *observed states*, *hidden (intrusion) states*, and *HMM profiles*. HMM training using initial data and continuous re-estimation creates profile that consists of transition probabilities and observation symbol probabilities. Steps involved in HMM modeling include:
(1) Measuring *observed states* that are analytically or logically derived from the intrusion indicators. These indicators are test-points spread all over the system.
(2) Estimating *instantaneous observation* probability matrix that indicates the probability of an observation, given a hidden state $p(S_i|O_i)$. This density function can be estimated using explicit parametric model (Multivariate Gaussian) or implicitly from data via non-parametric methods (multivariate kernel density emission).
(3) Estimating *hidden states* by clustering the homogeneous behavior of single or multiple components together. These states are indicative of various intrusion activities that need to be identified to the administrator.
(4) Estimating *hidden state transition* probability matrix using prior knowledge or random data. This prior knowledge and long term temporal characteristics are an approximate probability of state components transitioning from one intrusion state to another.

4. EMISSION STATES

Observed states represent competing risks derived analytically or logically using intrusion checkpoint indicators. Machine intrusion can be considered to be a result of several components competing for the occurrences of the intrusion. In this model intrusion checkpoint engine derives continues multivariate observation which is similar to the mean and standard deviation model except that it is based on correlations among two or more metrics. These observations $b_j(\boldsymbol{v})$ have continuous pdf and are a mixture of multivariate Gaussian (normal) distributions and expressed [9] as:

$$b_i(\boldsymbol{v}) = \sum_{k=1}^{M} c_{jk} \left(\frac{1}{(2\pi)^{M/2} |\boldsymbol{\Sigma}_{jk}|^{1/2}} exp\left(-\frac{1}{2} \left(\boldsymbol{v} - \boldsymbol{\mu}_{jk} \right)^T \boldsymbol{\Sigma}_{jk}^{-1} \left(\boldsymbol{v} - \boldsymbol{\mu}_{jk} \right) \right) \right) \quad (1)$$

where $(\cdot)^T$ denote transpose and

$$\begin{aligned}
c_{jk} &\geq 0 \text{ and } \sum_{k=1}^{M} = 1 \\
\boldsymbol{\Sigma}_{jk} &= \text{covariance matrix of the } k\text{-th mixture component of the } j\text{-th state} \\
\boldsymbol{\mu}_{jk} &= \text{mean vector of the } k\text{-th mixture component of the } j\text{-th state} \\
\boldsymbol{v} &= \text{observation vector} \\
M &= \text{number of dimensions of an observation with a multivariate Gaussian distribution} \\
\theta_{jk} &= (\boldsymbol{\Sigma}_{jk}, \boldsymbol{\mu}_{jk}) = \text{Gaussian components} \\
\eta_{jk} &= \text{drift factor of the } k\text{-th mixture component of the } j\text{-th state} \\
\lambda_{jk} &= (\theta_{jk}, c_{jk}, \eta_{jk}) = \text{user profile components.}
\end{aligned}$$

It is the responsibility of the intrusion checkpoint engine to re-estimate the λ_{jk} parameters dynamically for all matrices and all possible attack states. Various matrices that represent dimensions of an observation are the following:
(1) *Resource activity trend* is the measure of a resource activity that is monitored over a larger sampling period and has characteristics that repeat over that sampling period. For example, CPU activity changing depending upon the time of the day. Each period of activity can be though of as an extra dimension of activity measure.
(2) *Event interval* is a measure of an interval between two successive activities. For example, logging attempts between two successive intervals falls in this category.
(3) *Event trend* is the measure of events monitored over a larger sampling period with an objective to calculate the event behavior with a built-in repeatability. For example, the count of logging attempts in a day falls in this category.

5. HIDDEN INTRUSION STATES

Hidden states $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_{N-1}, \boldsymbol{S}_N\}$ are the set of states that are not visible but each state randomly generates a mixture of the M observations (or visible states O). The probability of the subsequent state depends only upon the previous state. Complete model is defined by the following probabilities; *transition probability matrix* $\boldsymbol{A} = \{a_{ij}\}$ where $a_{ij} = p(\boldsymbol{S}_i|\boldsymbol{S}_j)$, *observation probability matrix* $\boldsymbol{B} = (\boldsymbol{b}_i(\boldsymbol{v}_m))$ where $\boldsymbol{b}_i(\boldsymbol{v}_m) = p(\boldsymbol{v}_m|\boldsymbol{S}_i)$, and an initial probability vector $\boldsymbol{\pi} = p(\boldsymbol{S}_i)$. Observation probability represents an attribute that is observed with some probability if a particular failure state is anticipated. The model is represented by $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. Transition probability matrix is a square matrix of size equal to the number of states and represents the state transition probabilities. The observation probability distribution is a non-square matrix whose dimension equals the number of states by number of observables and represents the probability of an observation for a given state. The Intrusion Detection (IDS) as defined in this paper uses

the following states:
(1) *Normal* (N) state indicates the profile compliance.
(2) *Hostile intrusion attempt* (HI) indicates a hostile intrusion attempt that is in progress. This is typical of an external agent trying to bypass the system security.
(3) *Friendly intrusion attempt* (FI) indicates a non-hostile intrusion attempt that is in progress. This is typical of an internal agent trying to bypass the system security.
(4) *Intrusion in progress* (IP) indicates an intrusion activity that is setting itself up. This includes attempt to privileged resources, acceleration in resource usage etc.
(5) *Intrusion successful* (IS) indicates a successful intrusion. A successful intrusion will be accompanied with unusual resource usage (CPU, memory, IO activity, etc.).

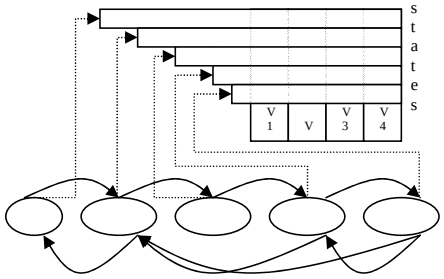

Figure 1: HMM model with five intrusion states and four Gaussian distributions for each state. Each Gaussian distribution can be represented as a mixture component of an observation.

6. IDS ARCHITECTURE

In ad hoc networks, an IDS system is deployed at the nodes to detect the signs of intrusion locally and independent of other nodes, instead of routers, gateways or firewalls.

In this section we will define components of the intrusion detection system that cooperate with each other to predict an attack state. After the model is trained, it enters a runtime state where it examines and classifies each valid observation. It then decides to either add it to a profile update, reject it, or mark it un-classified. This decision is important because a drift in the user's normal behavior may also represent an attack situation. An un-classified observation is monitored for classification in the future. This observation will later be rejected as a noise, or classified to a valid state based on the trending, similarity between unclassified states tending toward certain classification, and feedback from state machine based on other independent observations. Various components of an intrusion detection system are explained as follows:

6.1 Profile Estimator (PE)

The profile estimator is responsible for maintaining/re-estimating user profiles, classifying an observation to an attack state, triggering an alert upon detecting suspicious observation, or acting on the HMM feedback for re-estimation of profile. User profile data consist of pdf parameters represented by $\lambda_{jk} = (\mathbf{\Sigma}_{jk}, \boldsymbol{\mu}_{jk}, c_{jk}, \eta_{jk})$ where j represents the intrusion state and k represents the GMM mixture component. A new observation is evaluated against this profile which results in its classification and drift detection.

6.2 Instrumentation

Instrumentation produces an event data that is processed and used by *clustering agent* to estimate the profile. Component identification and measurements involve setup to ensure whether events should be sampled at regular intervals or be notified (or alerted) as an event vector upon recording some changes in pattern. The sensor data should be able to analyze data either as they are collected or afterwards and be able to provide real-time alert notification for suspected intrusive behavior. This will require fast acting silicon hooks that are capable of identifying, counting, thresholding, time-stamping, eventing, and clearing an activity. Examples of such hooks are performance counters, flip counters (or transaction counters), header sniffers, fault alerts (page faults etc.), bandwidth usage monitors etc. At the same time software instrumentation is also required to sample software related measurements like session activity, system call usage between various processes and applications, file-system usage, swap-in/swap-out usage, etc. Most of the operating systems support these hooks in the form of process tracking (e.g., PID in UNIX). The combination of these fast-acting hooks along with sampling ability are clustered together to enact an observation.

6.3 Data Clustering

Observation data are dependent upon the aggregation of events that are active. For example, a resource fault event generated by resource utilization engine is further categorized into *fault types* like page fault. Page faults count and *invalid page faults* in a sampled interval, represent instances of measurement (m_1, m_2). An observation (emission) can be a set of co-related measurements but represented by a single probability distribution function. Each of these measurement carries different weights as in multivariate Gaussian distribution. For example, disk I/O usage may be related to network I/O usage because of NFS. Such relationship is incorporated into the profile for the completeness of the observation and reduces the dimensionality for effective runtime handling. To achieve this reduced dimensionality, model uses the supervised self-organizing networks (sSOM) [10], which is a clustering method to map similar data to nearby function. HMM observation in this case is derived out of profile that represents a consolidated and single representation of NFS activity (sSOM output). A sample profile data structure is defined as follows.

NFS Profile {
  Observation Name = NFS Activity
  Input Events = {Disk I/O, Network I/O, $\cdots$}
  Output Emissions = Function (Input Events)
  PDF Parameter = $\{D[N], D[HI], D[FI], D[IP], D[IS]\}$
  Unclassified Observation = $\{U[t_1], U[t_2], \cdots, U[t_n]\}$
  Concept Drift Data = $\{\eta_{t1}, \eta_{t2}, \cdots\}$
}

6.4 Classifier

Observation data is analyzed for the purpose of sub classification to an appropriate attack state in a profile driven by different probability distribution parameters. Observation are also analyzed for concept drift to compensate for changes in user (or attack) behavior. Therefore, one of the objectives of the IC engine is to build classifier for j (attack states), that has the posterior probability $p(j|v)$ close to unity for one value of j and close to zero for all the others, for each realization. This can be obtained by minimizing the Shannon entropy given observed data v, which can be evaluated for each observation as

$$E = \sum_{v=1}^{M} p(j|v) \log(p(j|v)). \quad (2)$$

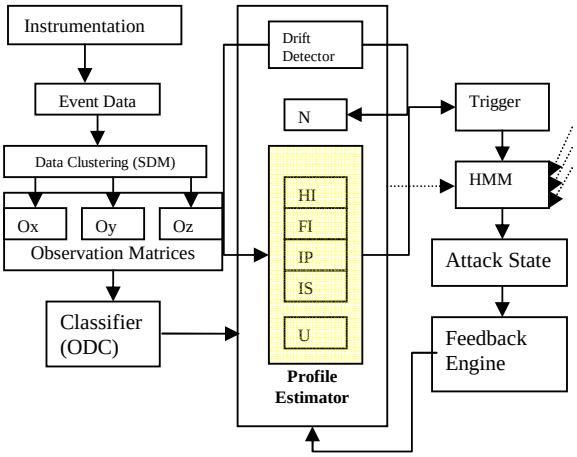

Figure 2: Intrusion checkpoint Engine - Re-estimation of the profile uses an observation classifier and an HMM feedback to the profile. Profile manager triggers an attention event if observation classifies to an attack state or cannot be classified (U). Attention event initiates an HMM state sequence prediction based on other continuous observations (dotted arrows) extracted in conjunction with profiles and state transition probabilities.

Each IC engine samples its observation independent from other observations (or emissions). Whenever it suspects an abnormal activity, it triggers an alert that causes an evaluation of most likely state using the Viterbi algorithm. As system changes its active behavior, the profile corresponding to that behavior is updated to avoid false positive evaluations by re-evaluating the model parameters using continuous estimation mechanism in real-time. New HMM parameters are evaluated again against the historical HMM parameters by comparing the entropy between old and retrained model. The expectation-maximization (EM) algorithm [11] provides a general approach to the problem of maximum likelihood (ML) parameter estimation in statistical models with variables that are not observed. The evaluation process yields a parameter set which it uses to assign observations points to new states. The computational complexity of the EM algorithm for GMMs is $O(i \times ND^2)$ where i is the number of iterations performed, N is the number of samples, and D is the state dimensionality. A common implementation choice is k-means algorithm in which k clusters are parameterized by their centroids with a complexity of $O(kND)$. A number of other algorithms can also be used of which X-means clustering [12] is one that reduces the complexity to $O(D)$.

6.5 Concept Drift Detector (CDD)

This module detects and analyzes the concept drifting [13] in the profile where training data set alone is not sufficient, and the model (profile) needs to be updated continually. When there is a time-evolving concept drift, using old data unselectively helps if the new concept and old concept still have consistencies and the amount of old data chosen arbitrarily just happen to be right [14]. This requires an efficient approach to data mining that helps select combination of new and old data (historical) to make an accurate re-profiling and further classification. The mechanism used is the measurement of *Kullback-Leibler (KL) divergence* [15], or relative entropy measures the kernel distance between two probability distributions of generative models. KL divergence is also the gain in Shannon information involved in going from the *a priori* to the *posteriori* expressed as

$$\alpha_{jkt} = KL(b, (v|\theta'_{jkt}), b_j(v|\theta_{jkt})) \quad (3)$$

where α_{jkt} is KL divergence measure, θ'_{jkt} is new Gaussian component, and θ_{jkt} is old Gaussian component of the k-th mixture of the j-th state at time t.

We can evaluate divergence by a Monte Carlo simulation using the law of large numbers [16] that draws an observation v_i from the estimated Gaussian component θ'_{jkt}, computes the log-ratio and averages this over M samples as

$$\alpha_{jk} \approx \frac{1}{M} \sum_{i=1}^{M} \log \left(\frac{b_j(v_i|\theta'_{jkt})}{b_j(v_i|\theta_{jkt})} \right). \quad (4)$$

KL divergence data calculated in the temporal domain are used to evaluate the speed of the drift, also called drift factor $0 \leq \eta \leq 1$. These data are then used to assign weights to the historical parameters that are then used for re-profiling.

6.6 Feedback Engine (FE)

This component is responsible for feeding back the current state information to the profile estimator. The current state information is calculated by running the Viterbi algorithm using the current pdf model parameters. This information is used by the EM algorithm that is a general method for improving the descent algorithm for finding the maximum likelihood estimation.

6.7 Relevant Profiles

In this section we will look into events that forms input to the profile structure. We define the HMM emissions as a processed observation derived from one or more temporal input events using a processor function. Exploiting temporal sequence information of event leads to better performance [17] of profiles that are defined for individual users, programs or classes. An abnormal activity in any of the following forms is an indicator of an intrusion or a worm activity:

(1) *CPU activity* is monitored by sampling faults, interprocessor interrupt (IPI) calls, context switches, thread migrations, spins on locks, and usage statistics.

(2) *Network activity* is monitored by sampling input error rate, collision rate, RPC rejection rate, duplicate acknowledgments (DUPACK), retransmission rate, timeout rate, refreshed authentications, bandwidth usage, active connections, connection establishment failure, header errors and checksum failures etc.

(3) *Interrupt activity* is monitored by sampling device interrupts (non-timer interrupts).

(4) *IO utilization* is monitored by sampling the I/O requests average queue lengths and busy percentage.

(5) *Memory activity* is monitored by sampling memory transfer rate, page statistics (reclaim rate, swap-in rate, swap-out rate), address translation faults, pages scanned and paging averages over a short interval.

(6) *File access activity* is monitored by sampling file access frequency, file usage overflow, and file access faults.

(7) *System process activity* is monitored by sampling processes with inappropriate process priorities, CPU and memory resources used by processes, processes length, processes that are blocking I/Os, zombie processes and command and terminal that generated the process.

(8) *System faults activity* represents an illegal activity (or a

hardware error) and is sampled to detect abnormality in the system usage. While rare faults represent a bad programming, but spurts of activity indicate an attack.
(9) *System calls activity* are powerful tools to get computer system privileges. An intrusion is accompanied with the execution of non-expected system calls. If the system-call execution pattern of a program can be collected before it is executed and is used to compare with the run-time system-call execution behavior, then non-expected execution of system calls can be detected. During real-time operation, a pattern-matching algorithm is applied to match on the fly the system calls generated by the process examined with entries of the pattern table. Based on how well the matching can be done, it is decided whether the sequence of system calls represents normal or anomalous behavior [18].
(10) *Session activity* is monitored by sampling the logging frequency, un-successful logging attempts, session durations, session time, and session resource usages, etc.

7. SYSTEM CONSIDERATIONS

Enormous amount of measurement data and computational complexity is an important consideration in the design of an effective IDS system. Various silicon hooks can be added to speed up the intrusion detection. In this section we describe some of these hooks:

7.1 Sensor data measurement (SDM)

hooks reduce the system complexity and increases the possibility of software reuse. SDM accelerates the combined measurements of the clustered components with an ability to send alerts using a systems policy. Hardware and software acts as a glue between transducers and control program that is capable of measuring the event interval, event trend (Section 4) with an ability to generate alerts on deviation from normal behavior (represented by system policy). The SDM hardware exists as a multiple-instance entity that receives alert vectors from various events spread all over the system. A set of correlated events that forms a cluster are registered against a common SDM instance. This instance represents the bayes optimal decision boundaries between set of pattern classes with each class represented by an SDM instance and associated with a reference vector. Each SDM instance is capable of trending and alerting and integrates the measurements from the event sensors into a unified view. Cluster trending analysis, is very sensitive to small signal variations and capable of detecting the abnormal signals embedded in the normal signals by supervised self-organizing maps [10] using learning vector quantization (LVQ). The strategy behind LVQ is to effectively train the reference vectors to define the Bayes Optimal decision boundaries between the SDM classes registered to an SDM instance.

7.2 Observation data classifier

(ODC) hooks accelerate the classification of an observation alert generated by SDM. This is a multiple-instance hardware capable of handling multiple observations in parallel. Each registered observation instance of the ODC hook consists of Gaussian probability distribution parameters of each state. Upon receiving an SDM alert, the observation corresponding to this alert is then classified to a specific state (Section 6.4). Re-classification of observed data may cause changes in the probability distribution parameters corresponding to the state. ODC is capable of maintaining the historical parameters which are then used to calculate concept drift properties (drift factor and drift speed, etc.) using KL drift detector.

7.3 GMM calculator

calculates the probability of the Gaussian mixture for each state, using the current observation. During the system setup, event vectors are registered against SDM instance (Fig. 3). These events are clustered and processed in its individual SDM. The processing includes trigger properties that initiates an observation. These observations then act as single-dimensional events that are registered to its ODC. Upon receiving the trigger, ODC performs re-classification of the observation (derived from trigger) and calculates the concept drift. It should be noted, that this hardware is activated upon a trigger by its parent.

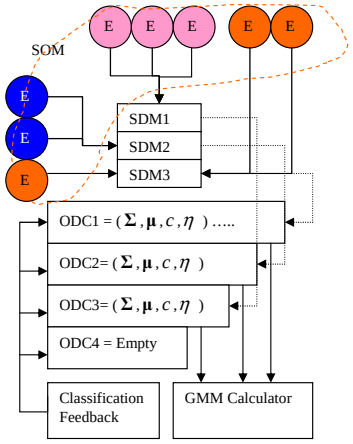

Figure 3: Illustration of the relationship between events (circles), sensors (SDM) and classifiers (ODC). Clusters of events (marked by similar colors) are registered to an SDM. SDM upon evaluating the event properties, generate an event to ODC. ODC is responsible for classification, trend analysis and drift calculation. Classification feedback acts as a feedback mechanism for re-estimation.

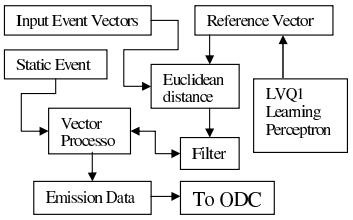

Figure 4: SDM instance functional block that uses LVQ1 clustering algorithm. Static events are created by routing an event vector and acts as principal trend setter.

8. CONCLUSION

As an experiment, we set up profile trainer for CPU activity, systems call activity, system process activity, network activity, and session activity (Section 6.7). All these profiles are related to the host activity for a specific user. This profile was trained on an Linux system over a period of 1 week on the activities related to these profiles. On the second week the same system then additionally ran subset of DARPA/99 [19] that contains newer and stealthy versions [20] of DARPA/98 attacks. We ran this training data for 2 weeks where the first week is attack free, and the second week is the attack run. We then ran subset of fourth and

fifth week of test data that contained 201 instances of about 56 types of attacks. We analyzed the performance of intrusion detection by measuring TRUE positives and FALSE positives based on the test data. We also measured the sensitivity of the training data by eliminating random days of the training sets. Fig. 5 displays ROC curves for the experiment using variable data sets and *Concept Drift factor* (Section 6.5) as the operating parameter. The false alarms rates stays low till the probability of detection reaches 90% . Further increase in detection probability (more than 90%), also increases the false positive rate. Detection probability remains constant till the False Positive probability reaches 35%, beyond which it increases slowly.

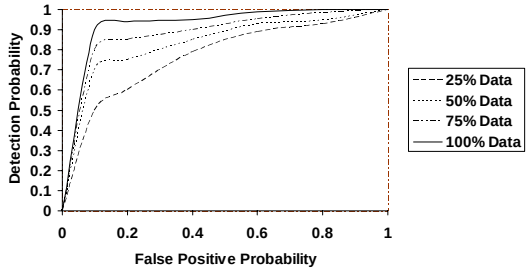

Figure 5: Receiver operating characteristics (ROC) curve showing the true positive detection/false positive ratio.

In ad hoc networks where wireless interface and MAC protocol make the node more prone to the attack, an exploit may use the weakest node to attack a network. Intrusion Detection by only monitoring deviations in network traffic patterns, restricted ports, protocol violations or retransmissions is not sufficient because certain abnormal-looking behavior may be valid because of media conditions and terrain. Certain attacks can work within threshold, and can only be detected by monitoring the affected components (like memory, CPU, login attempts, etc.). Since the intrusion state cannot be inferred directly by monitoring any specific parameters, we need to predict an attack based on mixture of observable data-points, events and current states. This leads to a statistical mechanism for intrusion prediction using HMM where observed data are represented as a weighted Gaussian mixture component. Using this mechanism, an observed deviation from a normal behavior carries a higher probability of being in an intrusion state. We also discussed a mechanism to distinctly identify an un-natural behavior from a concept drift due to the change in user's behavior and self-organizing networks mechanism to cluster all the correlated events in a single representation of an observation to an HMM model. This approach is helpful in detecting attacks as we attempt to identify the intruder's final intent after he has gained or is attempting to gain an illegal entry into the system instead of just detecting how he entered.

We also identified minimal hardware hooks that can accelerate intrusion detection data processing in real-time. While this is an effective approach to a successful intrusion detection, one of the shortcomings is the computational overhead that has to be mitigated using a custom ASIC to support IDS computations. This hardware has to be cost effective with low power consumption. We are currently evaluating cost/power/performance characteristics of a sample intrusion detection that uses buffer overflow attack to gain an illegal entry. Future work includes developing a mechanism to exchange the intrusion data with other members of an ad hoc network to identify a malicious node. This is required to prevent a compromised (or malicious) node participate in an ad hoc network, thereby preventing a large-scale attack due to cooperating nature of the member nodes. Also being investigated is an energy-efficient mechanism to limit the IDS activity during non-intrusive periods while accelerating it upon suspicious activity. We are exploring the possibility of using hybrid methods that uses HMM state feedback, event threshold, and random sampling. While this approach is very useful in detecting a ripple in the normal usage of a system, it is computationally expensive in terms of compute cycles and memory usage. Therefore this model is being enhanced to a hybrid HMM/DBN (Dynamic Bayesian Network) approach where HMM is used for modeling of temporal IDS characteristics while state probability model is represented by Bayes network.

9. REFERENCES

[1] E. Royer and C. K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, pp. 45–55, Apr. 1999.

[2] C. Ko, G. Fink and K. Levitt, "Automated detection of vulnerabilities in privileged programs by execution monitoring," in *Proc. 10th Annual Computer Security Applications Conference*, Orlando, FL, Dec, 134-144, 1994.

[3] D. Wagner and D. Dean, "Intrusion detection via static analysis," in *Proc. IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 2001.

[4] N. Ye, X. Li, Q. Chen S. M. Emran, and M. Xu, "Probabilistic techniques for intrusion detection based on computer audit data," *IEEE Trans. SMC-A*, vol. 31, pp. 266–274, Jul. 2001.

[5] S. Manganaris, M. Christensen, D. Serkle, and K. Hermix, "A data mining analysis of RTID alarms," *2nd International Workshop on Recent Advances in Intrusion Detection*, Purdue University, West Lafayette, Indiana, USA, Sep. 1999.

[6] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb. 1989.

[7] H. Bunke and T. Caelli,"Hidden Markov models: Applications in computer vision," *World Scientific, Series in Machine Perception and Artificial Intelligence*, vol. 45, 2001.

[8] F. Cuppens, "Managing alerts in a multi-intrusion detection environment," in *Proc. 17th Annual Conf. Computer Security Applications*, New Orleans, Louisiana 2001.

[9] A. Lee,"Gaussian mixture selection using context independent HMM," in *Proc. 2001 IEEE ICASSP*, 2001.

[10] T. Kohonen, *Self-organizing maps*. Springer Press, 1995.

[11] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 47-59, Nov. 1996.

[12] D. Pelleg and A. Moore, "Xmeans: extending K-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Machine Learning*, p. 727, 2000.

[13] G. Widmer and M. Kubat, "Learning in the presence of concept drifting and hidden contexts," *Machine Learning*, vol. 23, pp. 69–101, 1996.

[14] W. Fan, "Systematic data selection to mine concept-drifting data streams," *ACM SIGKDD*, 2004.

[15] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, Mar. 1951.

[16] G. R. Grimmett and D. R. Stirzaker, *Probability and random processes*. Oxford, U.K.: Clarendon Press, 2nd edition, 1992.

[17] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *Proc. Workshop on Intrusion Detection and Network Monitoring*, pp. 51–62, Santa Clara, USA, Apr. 1999.

[18] A. Wespi, H. Debar, and M. Dacier, "An intrusion-detection system based on the Teiresias pattern-discovery algorithm," *Eicar'99*, Aalborg, Denmark, Feb. 27-Mar. 2, 1999.

[19] DARPA, "DARPA 1999 Intrusion Detection Evaluation," *http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html*, 1999.

[20] R. P. Lippmann and R. K. Cunningham, "Guide to creating stealthy attacks for the 1999 DARPA off-line intrusion detection evaluation," *MIT Lincoln Laboratory Project Report IDDE-1*, June 1999.