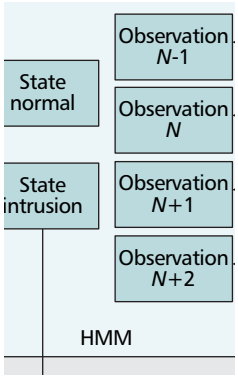
CONTROL THEORETIC APPROACH TO INTRUSION DETECTION USING A DISTRIBUTED HIDDEN MARKOV MODEL

RAHUL KHANNA, INTEL CORPORATION
HUAPING LIU, OREGON STATE UNIVERSITY



Cooperative ad hoc wireless networks are more vulnerable to malicious attacks than traditional wired networks. The authors discuss a control-theoretic hidden Markov model (HMM) strategy for intrusion detection using distributed observation across multiple nodes.

ABSTRACT

Cooperative ad hoc wireless networks are more vulnerable to malicious attacks than traditional wired networks. Many of these attacks are silent in nature and cannot be detected by conventional intrusion detection 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 normal behavior. In this article we discuss a control-theoretic hidden Markov modelstrategy for intrusion detection using distributed observation across multiple nodes. This model comprises a distributed HMM engine that executes in a randomly selected monitor node and functions as a part of the feedback control engine. This drives the defensive response based on hysteresis to reduce the frequency of false positives, thereby avoiding inappropriate ad hoc responses.

INTRODUCTION

An intrusion detection system (IDS) protects data integrity and manages system availability during an intrusion. In a mobile ad hoc network (MANET) with self-regulating properties [1] it must deal with challenges related to resource-constrained fully mobile, self-configuring multihop wireless networks with varying resources and limited bandwidth. This mechanism should be able to detect intrusion by monitoring unusual activities in the system and comparing them 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. The distributed and cooperative nature of ad hoc network nodes makes it possible for a malicious node to exploit the weakest node by hijacking it or launching an attack through it. This inherent vulnerability can disable the whole network cluster and further compromise security by impersonating, message contamination, hijacking, passive listening, or acting as a malicious router. Various routing techniques have been researched in this area of trying to resist attacks [2]. Intrusion can be thought of as a pattern of an observed sequence. Its detection is similar to an immune system that identifies and eliminates anomalies by measuring deviations from normal processes using distributed identifiers over the system with an identifiable and adaptable relationship. This can be supported using a model where each state has probabilistic distribution of producing identifiable observations and a transition matrix to other states.

A hidden Markov model (HMM) [3] is one such model that correlates observations (parameters changes, fault frequency, etc.) [4] to predict hidden states that factor in the system design. Observation points are optimized using an acceptable set of system-wide intrusion checkpoints (ICs) while hidden states are created using explicit knowledge of probabilistic relationships with these observations. For modeling a large number of temporal sequences, HMM acts as an excellent alternative, as it has been widely used for pattern matching in speech recognition and image identification. Some of the previous work on IDS using HMM includes an HMM-based predictive model capable of discriminating between normal and abnormal behaviors of network traffic [5], a framework for handling multiple sensors implemented by representing each of the sensors monitoring a host with an HMM [6], HMM-based detection of complex Internet attacks consisting of several steps that occur over an extended period of time [7], HMM-based anomaly decisions at the system call level using sequences of system call traces as observable [8], and HMM-based algorithms for building behavior classifiers capable of detecting intrusion attempts on computer systems [9]. Other work in this area includes a statistical approach [10] that monitors the system

1536-1284/08/$25.00 © 2008 IEEE

call trace of a program's execution for compliance with the precomputed model and an alert-based policy mechanism [11] that associates an alert with multiple events frequently occurring together.

In this article we extend the traditional HMM-based IDS approach by using a control-theoretic distributed HMM to make it suitable for ad hoc networks in which nodes could have very limited power and processing capabilities. The basic ingredient of this approach is the proportional-integral differential (PID) control engine and the distributed HMM processor on a randomly selected monitor node. While the PID control engine drives the defensive response based on feature hysteresis to reduce the frequency of false positives, distributed HMM processing distributes the computational load of training and state estimation among member nodes of the ad hoc node cluster. Monitor nodes are selected periodically and randomly using a cost function that favors long-term relationships, average battery conditions, estimation trends, and fair loading. PID controllers do not require advanced mathematics to characterize the model underlying the checkpoint measurements and can easily be adjusted to the desired response. These controllers can easily be implemented as silicon hooks coupled to the monitored intrusion checkpoints.

IDS Modeling Ingredients

The objective of the modeling scheme is to identify the intrusion while reducing the number of false positives. An instantaneous deviation from a normal profile can be construed as an intrusion due to a momentary change in the system environment. Such deviations may be legal, as also seen during installation of new patches in operating systems. Therefore, in an IDS we have to fulfill multiple objectives related to accurate intrusion detection using various ingredients like:

Intrusion checkpoints to analyze the sensor activity that predicts the transition from normal state to an intrusion state. Intrusion checkpoints also represent the observable states of the IDS.

Creation of an activity profile that identifies abnormal activity of the observable states by measuring the sensor deviation from normal behavior. It 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).

Concept drift that measures the change in user behavior over a period of time.

Control loop that adapts the IC trigger based on the weighted sum of proportional, average, and derivative sensor measurements over derivative and integral time window.

Model that predicts the most probable state based on previous state (normal/intrusion) as well as observed states. This can be accomplished using the HMM, as described later in this section.

Hidden Markov Model

HMM-based approaches correlate the system observations (usage and activity profile) and state transitions to predict the most probable *intrusion state sequence*. An HMM is a stochastic model 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*. The states of the HMM can only be inferred from the observed symbols; hence, it is called *hidden*. HMM modeling schemes consist of *observed (intrusion checkpoints) states*, *hidden (intrusion) states*, and *HMM (activity) profiles*. HMM training using initial data and continuous re-estimation creates a profile that consists of transition probabilities and observation symbol probabilities. Steps involved in HMM modeling include:

Measuring *observed states* that are analytically or logically derived from the intrusion indicators. These indicators are test points spread all over the system representing competing risks derived analytically or logically using IC indicators. Machine intrusion can be considered to be a result of several components competing for the occurrences of the intrusion. In this model an IC engine derives continuous multivariate observation, which is similar to the mean and standard deviation model except that it is based on correlations among two or more metrics:

- Resource activity trend: the measure of a resource activity monitored over a larger sampling period, it has characteristics that repeat over that sampling period. For example, CPU activity changes depending on the time of the day. Each period of activity can be thought of as an extra dimension of activity measure.
- Event interval: a measure of an interval between two successive activities. For example, logging attempts between two successive intervals fall in this category.
- Event trend: the measure of events monitored over a large 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.

Estimating an *instantaneous observation* probability matrix that indicates the probability of an observation, given a hidden state $p(\boldsymbol{S}_i|\boldsymbol{O}_i)$. This density function can be estimated using an explicit parametric model (multivariate Gaussian) or implicitly from data via nonparametric methods (multivariate kernel density emission).

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. 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 randomly generates a mixture of the M observations (or visible states O). The probability of the subsequent state depends only on the previous state.

Estimating a *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.

The complete HMM 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* $\boldsymbol{B} = (b_i(v_m))$, where $b_i(v_m) =$

The objective of the modeling scheme is to identify the intrusion while reducing the number of false positives. An instantaneous deviation from a normal profile can be construed as an intrusion due to a momentary change in the system environment.

While integral response measures the amount of time the error has continued uncorrected, differential response anticipates the future errors from the rate of change of error over a period of time.

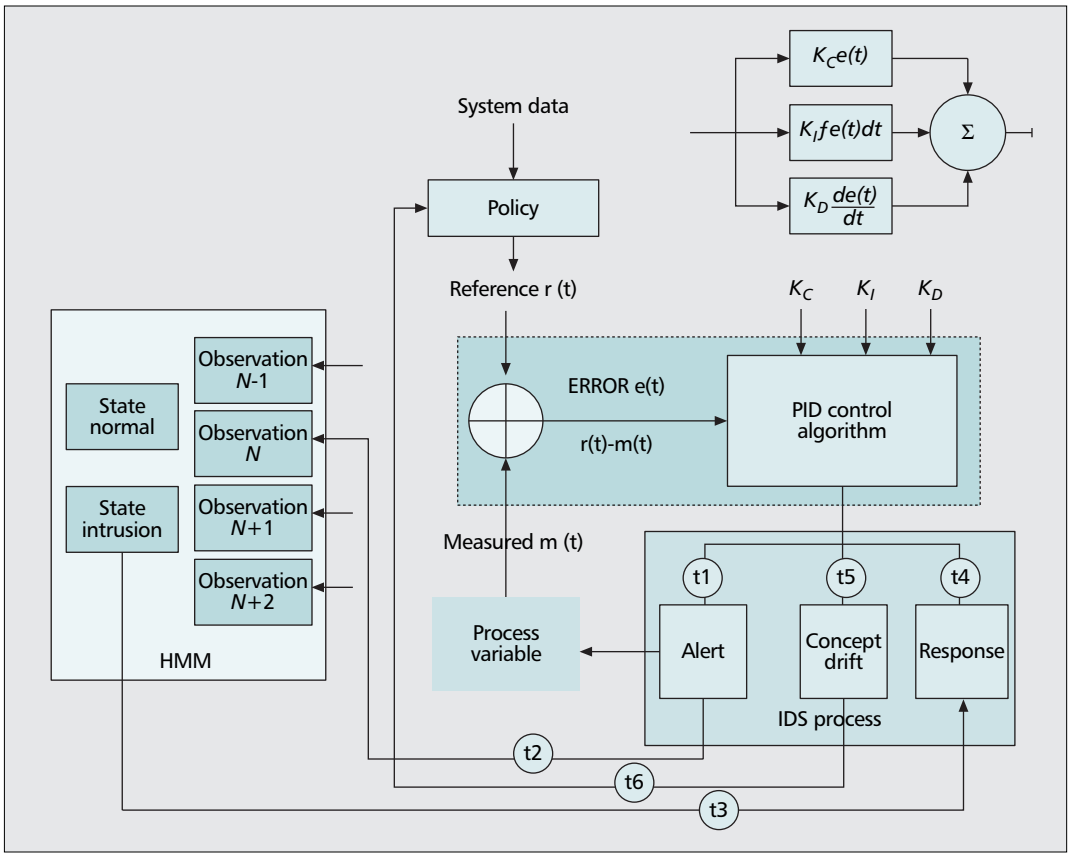


Figure 1. *PID control loop for intrusion checkpoint. The process output (alert) constitutes the observation (emission) in an HMM. A true-positive response is fed back to the process response unit of the PID control to aid runtime retraining. Concept drift analysis aids in resetting the reference point.*

$p(v_m|\boldsymbol{S}_i)$, and an initial probability vector $\pi = p(\boldsymbol{S}_i)$. The observation probability represents an attribute that is observed with some probability if a particular failure state is anticipated. The model is represented by $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{B}, \pi)$. The 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 the number of observable states, and represents the probability of an observation for a given state. The IDS has the following states:

- Normal (N) state indicates profile compliance.
- Intrusion in progress (IP) indicates an intrusion activity that is setting itself up. This includes attempts to gain privileged resources, acceleration in resource usage, and so on.
- Intrusion successful (IS) indicates a successful intrusion. A successful intrusion will be accompanied with unusual resource usage (CPU, memory, IO activity, etc.).

Intrusion Checkpoint Control

In this section we discuss the applicability of the control-theoretic architecture shown in Fig. 1 that drives a defensive response based on hysteresis to reduce the frequency of false positives, thereby avoiding inappropriate ad hoc responses. Excessive responses can slow down the system and negatively impact the effectiveness of the IDS. IDS control responses are related to adjusting component functionality (e.g., throttling), alert generation (to predict intrusion state), and analyzing concept drift. An appropriate response is built into the IC process that predicts the attack pattern and triggers the selective response in a PID control. The PID controller takes a measured value from an IC and compares it to a reference value. The difference is then used to trigger an alert (abnormal activity) to the process in order to bring the process' measured value back to its desired setpoint. The PID controller can adjust the process outputs based on the history and rate of change of the error signal, which gives more accurate and stable control. This avoids a situation where alerts may not be true representations of intrusion activity due to false positives. Such miscalculations can result in either disproportionate and costly defensive measures or complete security failure. It is therefore essential to build a weighted integral and differential response to the trigger mechanism instead of reacting to an instantaneous measurement. While an integral response measures the amount of time the error has continued uncorrected, differential response anticipates future errors from the rate of change of error over a period of time. The reference (setpoint) values are dynamic in nature and set as a part of coarse-grained settings that are estimated over long periods of time. These re-estimates are required to account for changing user behavior, also referred to as *concept drift*.

The checkpoint control loop is the first state of a multistage sequential IDS. The process out-

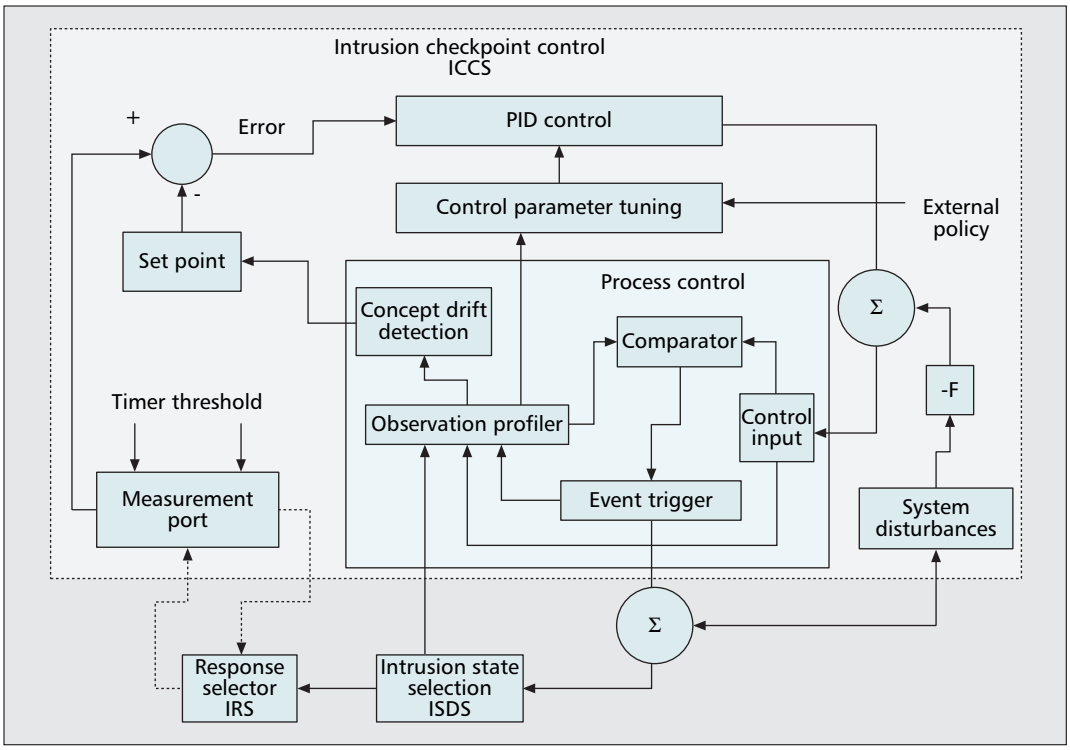


Figure 2. *ICCS is responsible for providing the stable observability data to the intrusion state detection stage. This data is profiled for variances due to changing user behavior and temporary changes in system environment (also referred to as disturbances).*

In ad hoc networks, an IDS is deployed at the nodes to detect the signs of intrusion locally and independent of other nodes, instead of routers, gateways or firewalls. The IDS architecture comprises of multiple stages with information feedback mechanism between stages.

put of the control loop provides observability of an individual IC that aids in state estimation. Collective observations from multiple checkpoints are fed into the statistical model (in this case HMM) responsible for predicting the state transition. It is imperative that any such output should be stable and free of oscillations. Response measures are delayed to account for delay involved in estimation of intrusion state based on observations from other checkpoints. While the setpoint (reference) is constant over a long period of time, it can change due to user behavior of system policy driven by a temporary change in the operating environment. System data and the process feedback provide hints that are then used to change the setpoint (or setpoint weights) in steps based on system policy. System policy is driven by long-term hysteresis based on the system's behavior and the well-known relationship with various checkpoints.

IDS Architecture

In this section we define components of the IDS that cooperate with each other to predict an attack state. In ad hoc networks an IDS is deployed at the nodes to detect signs of intrusion locally and independent of other nodes, instead of at routers, gateways, or firewalls. The IDS architecture comprises multiple stages with information feedback mechanismd between stages. These stages can be roughly defined as:

- Intrusion checkpoint control stage (ICCS), shown in Fig. 2, is the observability stage with an objective to produce stable emissions using continuous estimations. This stage is also responsible for detecting temporary changes due to legal activity and concept drift, signifying changing long-term user behaviors. This decision is important because a drift in the user's normal behavior may also falsely predict an attack situation. Observation can be rejected as noise, or classified to a valid state based on trending, similarity between unclassified states tending toward certain classification, and feedback from a state machine based on other independent observations.
- Intrusion state detection stage (ISDS) receives the observability data from multiple checkpoints and predicts the transition to one of the hidden states (normal, intrusion) based on a trained statistical model. An estimated intrusion decision is fed back to ICCS, which helps re-estimate the usage trends while avoiding any false positive preemptive responses.
- Intrusion response stage (IRS) is responsible for initiating the corrective (healing) actions due to state transition. These actions may scale back any abnormal activity as seen in the observability data. A mispredicted state transition may initiate an inappropriate response and will have a negative effect on checkpoint activity.

After various components of the model are trained, it enters a runtime state where it examines and classifies each valid observation. Various components of an IDS are explained in the following subsections.

Intrusion Checkpoint Control Stage

The ICCS represents the feedback control component for an individual IC. It comprises a measurement port, a PID controller, an observation profiler, a concept drift detector (CDD), and a feedback path to the process input.

Successive observations are evaluated against this profile which results in its new profiles and drift detection. An observation (emission) can also be a set of correlated measurements but represented by a single probability distribution function.

The measurement port comprises fast acting software and silicon hooks that are capable of identifying, counting, thresholding, timestamping, 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, session activity, system call usage between various processes and applications, file system usage, and swap-in/swap-out usage. Measured data is analyzed as it is collected or afterward to provide real-time alert notification for suspected intrusive behaviors. These fast acting hooks are clustered to enact an observation. Measurements can be sampled at regular intervals or cause an alert based on a user-settable threshold.

The observation profiler monitors various inputs for maintaining/re-estimating an activity profile that ascertains a rough (partially perfect) boundary between normal and abnormal activity characterized in terms of a statistical metric and model. A metric is a random variable representing a quantitative measure accumulated over a period. Measurements obtained from audit records used together with a statistical model analyze any deviation from a standard profile. The observation profiler receives multiple feedbacks from the PID control output, event trigger, and ISDS, and performs recursive estimations that generate successive probabilistic profile data estimates with closed-form solution. Trigger activity is generally followed by a change in the PID control output that initiates a recovery response. A true positive recovery response will scale back the checkpoint activity to normal. A false positive action will instead cause oscillations, degraded system performance, or little change in the measured error. Activity profile data consists of probability distribution function (pdf) parameters represented by $\lambda_j = (\sigma_j, \mu_j, \eta_j)$, where σ_j, μ_j, and η_j represent variance, mean, and activity drift factor, respectively. Successive observations are evaluated against this profile, which results in its new profiles and drift detection. An observation (emission) can also be a set of correlated measurements but represented by a single pdf. Each of these measurements carries different weights as in multivariate probability distribution. Such a relationship is incorporated into the profile for completeness of the observation and reduces the dimensionality for effective runtime handling. Observability in this case is derived out of the profile that represents a consolidated and single representation of activity. A sample profile data structure is defined as follows:

```
NFS Activity Profile {
    Observation Name = NFS Activity
    Input Events = {Disk I/O, Network I/O, ···}
    Observation Trigger = Function (Input Events)
    PDF Parameter = {D[N], D[IS]}
    Unclassified Observation = {U[t1], U[t2], ···, U[tn]}
    Concept Drift Data = {ηt1, ηt2, ···}
}
```

The concept drift detector detects and analyzes concept drifting [12] in the profile where the 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 and old concepts still have consistencies, and the amount of old data chosen arbitrarily just happens to be right [13]. This requires an efficient approach to data mining that helps select a combination of new and old (historical) data to make accurate reprofiling and further classification. The mechanism used is the measurement of Kullback-Leibler (KL) divergence [14], 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 a priori to *a posteriori*, expressed as

$$\alpha_t = KL(b(v|\theta_t'), b(v|\theta_t)), \qquad (1)$$

where α_t is KL divergence measure, θ_t' is the new Gaussian component, and θ_t is the old Gaussian component at time t.

We can evaluate divergence by a Monte Carlo simulation using the law of large numbers [15] that draws an observation v_i from the estimated Gaussian component θ_t', computes the t log-ratio, and averages this over M samples as

$$\alpha_t \approx \frac{1}{M}\sum_{i=1}^{M} \log\left(\frac{b(v_i|\theta_t')}{b(v_i|\theta_t)}\right). \qquad (2)$$

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

The feedback path is responsible for feeding back the current state information to the profile estimator. The current state information is calculated by running the ISDS module using the current model parameters. This information is then used by the profiler to filter out any noise and re-estimate the activity profile data. If a trigger activity is not followed by a state transition, a corrective action is performed to minimize the false positives in the future.

The PID controller generates an output that initiates a corrective response applied to a process in order to drive a measurable process variable toward a reference value (setpoint). It is assumed that any intrusion activity will cause variations in the checkpoint activity, thereby causing a large error. Errors occur when a disturbance (intrusion) or load on the process (changes in environment) changes the process variable. The controller's mission is to eliminate the error automatically. The discrete form of PID controller is represented by the following equation:

$$u(nT) = K_p e(nT) + K_i T \sum_{i=(nT-w)}^{nT} e(i) + K_d \frac{e(nT) - e(nT-1)}{T} + u_0, \qquad (3)$$

where $e(t)$ is the error represented by the differ-

ence between measured value and setpoint, w is the integral sampling window, nT is the nth sampling period, and K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively.

Stability is ensured using the proportional term, the integral term permits the rejection of a step disturbance, and the derivative term is used to provide damping or shaping of the response. The desired closed-loop dynamics are obtained by adjusting these parameters iteratively by *tuning* and without specific knowledge of an intrusion detection model. Control parameters are continuously tuned to ensure the stability of the control loop in a control-theoretic sense, over a wide range of variations in the checkpoint measurements. While control parameters are evaluated frequently, they are updated only when improvement in stability is anticipated. These updates can be periodic over a long period of time.

Intrusion State Detection Stage

The ISDS defines the statistical model responsible for predicting the current intrusion state based on observability data received from ICCS modules. In this context we choose HMM as described earlier. The HMM states are hidden and indirectly evaluated based on model parameters. ICCS trigger output acts as an emission to a specific HMM model and allocates a weight according to their confidence. Observation probability is expressed as a mixture of individual observation probabilities from multiple checkpoints and improves the performance of an IDS. A mixture model is a model in which the independent variables are measured as fractions of a total. The mixture model can be represented as

$$p(x) = \sum_{k=1}^{K} a_k h(x|\lambda_k), \quad (4)$$

where $p(x)$ is the modeled probability distribution function, K is the number of components in the mixture model, and a_k is the mixture proportion of component k.

Observations from multiple checkpoints are distributed among a mixture of models with weights given to each model based on trivial knowledge and continuous training. This approach is advantageous as it allows one to model the intrusion states at varying degrees of granularity while retaining the advantages of each model. Depending on the data characteristics (amount of data, frequency), models can be adapted by modifying weights such that complex models are favored for complex inputs and vice versa.

In Fig. 3 the HMMx subblock is responsible for receiving the *abnormal activity* alert and processes the interrupt to service the hidden state (intrusion) estimation. It maintains the *HMM data* and interacts with the expectation-maximization (EM) block and state-estimation (SE) block for retraining and state prediction flows. This block also implements reduced dimensionality by combining multiple inputs into a single observation with its own pdf. This observation is then fed into the EM and SE blocks for state estimation.

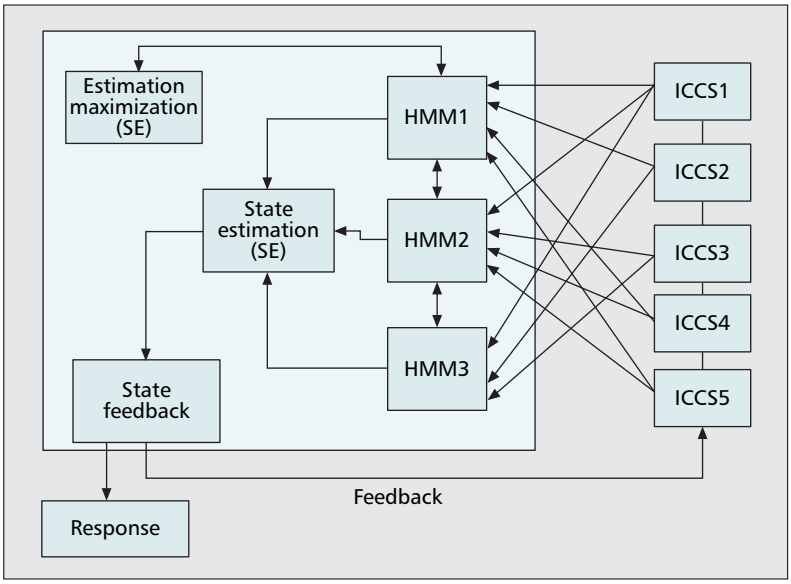


Figure 3. *Intrusion state detection stage.*

The EM algorithm [16] 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 it uses to assign observations points to new states. The EM subblock is responsible for finding the ML estimates of parameters in the HMM model as well as mixture densities (or model weights) and relies on the intermediate variables (also called latent data) represented by state sequence. EM alternates between performing an E-step, which computes an expectation of the likelihood, and an M-step, which computes the ML estimates of the parameters by maximizing the expected likelihood found on the E-step. The parameters found on the M-step are then used to begin another E-step, and the process is repeated. In the HMM mixture modeling, intrusion checkpoint events under consideration have membership in one of the distributions we are using to model the data. The job of estimation is to devise appropriate parameters for the model functions we choose, with the connection to the data points being represented as their membership in the individual model distributions.

SE is responsible for modeling the underlying state and observation sequence of the HMM mixture to predict state sequences for new intrusion states using the Viterbi algorithm (to find the most likely path through the HMM). A trained mixture appears to be a single HMM for all purposes and can be applied as a standard HMM algorithm to extract the most probable state sequence given a set of observations. Estimates for the transition and emission probabilities are based on multiple HMM models and are transparent to the standard HMM models. The Viterbi algorithm is a dynamic algorithm requiring time $O(TS^2)$ (T is the number of time steps and S is the number of states) where at each time step it computes the most probable path for each state given that the most probable path for all previous time steps has been computed. The state feedback subblock feeds back the estimated

Selecting an appropriate response for different intrusions cannot be handled by a simple policy decision. This requires a complete understanding of active intrusion responses which is still an open problem. An over-reactive response can turn into a denial of service (DoS) attack.

state to the *observation profiler* in ICCS (Fig. 2), which then uses this data for recalibrating the profile.

As part of the proactive approach in an active IDS, the response unit encapsulates various actions that are undertaken upon a suspected intrusion. These actions can be automated or manual based on complexity and prior knowledge. The response unit modifies the state of the attacked system to thwart or mitigate the effects of the attack. Such control can take the form of terminating network connections, increasing security logging, killing errant processes, APR poisoning, using decoys (false IP address), and so on. This action is also important because after raising the abnormal activity alert, the profiler (ICCS) constantly monitors the abnormal activity (PID control output) and expects it to be reduced based on some external actions (automated or manual). This action is equivalent to the process control function that influences the process variable in the feedback control system with an objective to reduce the abnormal activity. Selecting an appropriate response for different intrusions cannot be handled by a simple policy decision. This requires complete understanding of active intrusion responses, which is still an open problem. An overreactive response can turn into a denial of service (DoS) attack.

Intrusion Detection in an Ad Hoc Network

Unlike wired networks, MANETs present new challenges to mobile hosts that are resource-constrained and have limited battery power. Additionally, these mobile nodes have inherent vulnerabilities due to lack of fixed infrastructure, cooperative algorithms, dynamically changing topology, and open physical access. Some of the common attacks that exploit these limitations are route messages and route information tampering, selective forwarding, sybil attack, sinkhole attack, wormhole attack, spoofing, packet flooding, packet dropping, location exposure, sleep deprivation (battery exhaustion), and radio jamming (medium access layer [MAC] layer attack). Adding to the problem, constantly changing topologies, an open medium, and volatile physical environments make it difficult to discriminate between an intrusion and normal operation. Intrusion detection thus requires extensive evidence gathering and comprehensive analysis. HMM provides such a mechanism and also overcomes the limitations of a signature-based technique of detecting unknown attacks by identifying changes in the system dynamics. Still, an HMM-based mechanism is computationally intensive and requires mechanisms to reduce the algorithmic overhead. We overcome this shortcoming by distributing the computational load among member nodes of MANETs that share a similar environment and operating conditions due to proximity.

In this section we discuss a cooperative IDS that involves participation of the member nodes in a global decision process. This involves distributed processing among local nodes and randomly elected monitoring nodes. While the ICCS is implemented locally using silicon and software hooks, ISDS operations execute on the monitoring nodes. *Monitoring nodes* are at a single-hop distance and elected randomly at periodic intervals using a fairness and risk cost evaluation. Various factors such as the number of refusals, membership period, and voting patterns are considered for making this evaluation. While local nodes contribute trigger data locally and externally, monitor nodes consume this data to estimate the intrusion state through the contribution of observations from all member nodes. Whenever suspected activity is detected, it initiates an intrusion detection event that is propagated to monitor nodes. Monitor nodes in turn request sharable observation data from individual nodes. Based on multiple observations with node-level dimensionality, an HMM mixture algorithm is executed to predict the possible intrusion state.

IDS Node

Intrusion detection in a mobile local host is limited to profiling its local activity using floating ICCS modules. The intent is to reduce the system complexity and the possibility of software reuse. These hooks are presented to accelerate the combined measurements of the clustered components with an ability to send alerts based on a system-level policy. It contains the hardware and software that act as glue between transducers and a control program capable of measuring the event interval and event trend with an ability to generate alerts on deviation from normal behaviors (represented by system policy). In this specific case, the feedback control loop is implemented partially in the silicon (ICCS block) with configurable control parameters (Fig. 2). To further enhance auto-discoverability, modularity, and reusability, configuration and status registers are mapped into the capability pointer of the PCI express configuration space. Similar mechanisms exist today in very basic form as performance counters (PerfMon), leaky bucket counters, and so on. These counters need to be coupled with ICCS modules that contain the PID controller, profilers, threshold detectors, drift detectors, and coarse-grained tuners. ICCS modules are implemented in isolation from the measured components such that a single ICCS component can multiplex between multiple measurement modules. While some of the checkpoints are used for local consumption, others are shared with the monitor nodes to aid in cooperative state estimation. Examples of such checkpoints are *packet drop rate*, *route request rate*, and *route reply rate*. These checkpoints share the trigger data with the monitor nodes and constitute the node's contribution to the mixture of HMMs.

IDS Monitor

An IDS monitor is required to offload from the member nodes performing redundant computation by distributing the computational load and selecting a new monitor periodically. The monitor function is performed by sharing individual nodes' observations by either exchanging data or overhearing the node traffic on all the members

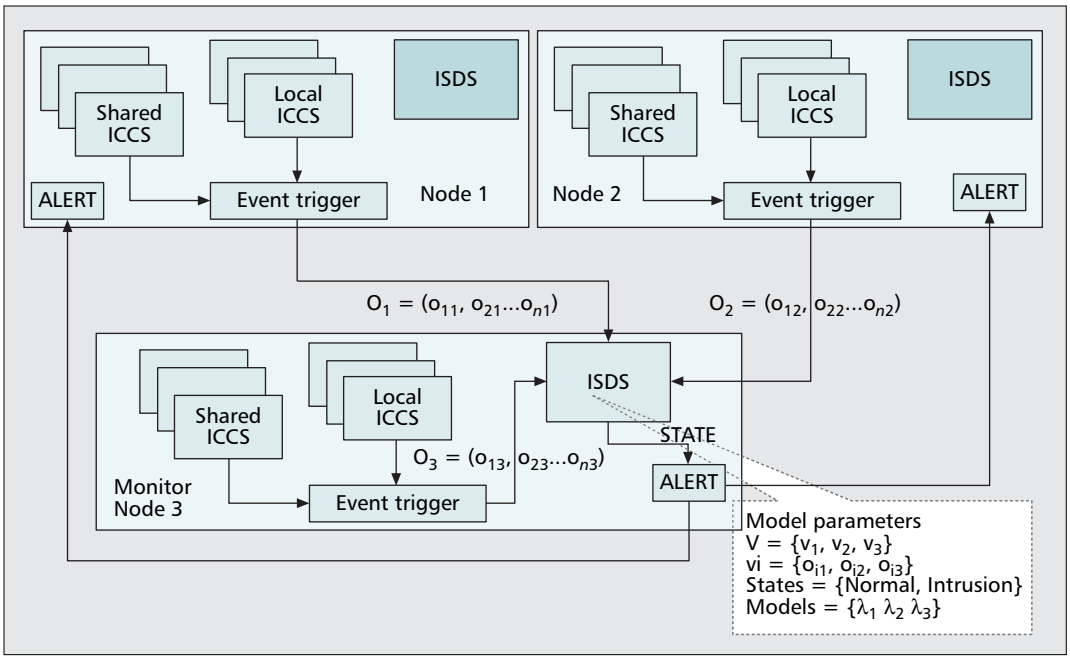


Figure 4. *Distributed state estimation using HMM. Similar observations from each node act as a combined observation at the monitor node according to its weight. Each node contributes multiple observations to the monitor node. The monitor node then executes the HMM mixture algorithm and returns the estimated state back to the host.*

Cooperative IDS provides us with not only with a lower battery consumption, but also with a hierarchical approach where local abnormalities are substantiated using shared processing among the member nodes of the ad hoc cluster. This evolves into an IDS tree.

of the cluster (Fig. 4). Some static information related to route and location still needs to be transmitted to the monitors. There can be more than one monitor in the cluster executing independent of each other and yielding this role to another node on cost evaluation. This allows accurate evaluation using multiple samples. Node monitors implement the HMM mixture model using multiple observations from all member nodes as defined by ISDS (Fig. 3). The monitor node is responsible for:

- Estimating the current state of the cluster (state estimation)
- Retraining the model based on cluster dynamics (expectation maximization)
- Initiating a trigger response to allow all member nodes to update any sharable information (location, routes, trigger data, etc.)
- Listening to member nodes who may be experiencing abnormal activity
- Yielding the monitor role to another monitor using a handoff mechanism
- Alerting the nodes of a change in intrusion state

Hence, the node monitor completes the feedback loop by initiating the response action in case the state transitioned to an intrusion state. It is expected that the response action will help scale back abnormal activity to normal activity and therefore reduce the control feedback error. In case of multiple monitors, each monitor votes on the estimated state, and the majority vote prevails. Cooperative IDS provides us not only with lower battery consumption, but also with a hierarchical approach where local abnormalities are substantiated using shared processing among the member nodes of the ad hoc cluster. This evolves into an IDS tree where host nodes act as leaf structures, and the monitor nodes act as the node structure with a cooperative decision process. These decisions can be accepted/rejected according to the host node's local policy.

Experimental Result

As an experiment, we set up an IC for received signal strength (RSS), round-trip time (RTT), bandwidth, and rate of packet drop on three mobile clients (laptops) running on an 802.11g wireless controller (Fig. 4). These clients are authenticated using SSL and kept stationary for experimental purposes. These clients exchange among themselves a 2-Mbyte training sequence periodically that is fragmented at tunable intervals with client 3 always acting as a monitor. This tunes the PID controller, profiler, and CDD for nominal operating conditions for the given condition. Additionally, the model consists of a traffic generator, an environment disrupter (changing signal strength), and an attack module that simulates different types of attacks. The traffic generator simulates real-time audio/video and TFTP traffic under random disruption. Upon event trigger by ICCS, all nodes transmit RSS, RTT, bandwidth, and packet drop rate data to the monitor node (node 3), which executes the HMM mixture model and returns the estimated status. As part of the recovery action, the attack is scaled back upon positive intrusion detection to allow the feedback control loop error to converge to set-point. Figure 5 shows the ROC characteristics under attack conditions by varying the sampling period parameter. In a disruptive environment (simulated by changing signal strength), the results are substantially better with a positive detection rate of 83–89 percent for sample intervals of 33–36 s.

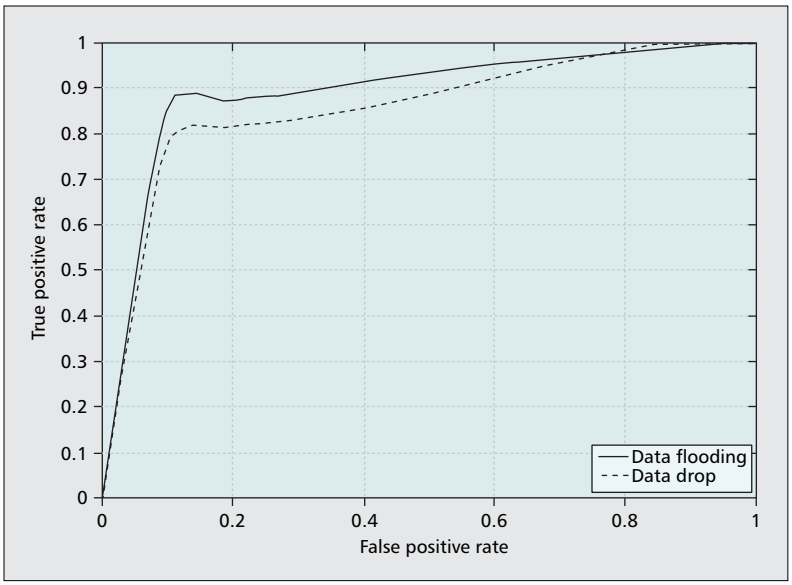


Figure 5. *Receiver operating characteristics (ROC) curve showing the true positive detection/false positive ratio.*

Traditional approaches like signature detection are very efficient for known attacks and reduce the number of false positives, but are incapable of detecting new or modified attack patterns. HMM-based techniques [4–8] solve this problem to a greater extent, but suffer from computational overhead [17]. In addition, traditional approaches lack the feedback mechanism that can correlate the corrective response to changes in system dynamics. Moreover, wireless mobility presents a unique case where redundancy in the environment can be exploited by reducing the redundant computation between ad hoc nodes. The new approach tries to solve these problems in a single model that uses HMM with distributed emissions between member nodes and a PID-based feedback control loop. In the absence of a PID controller, the false positive rate increases between 10–13 percent because of premature responses to transients.

Conclusion

While intrusion prevention may be the first line of defense, it is not foolproof. An exploit may use the weakest link (or node) to attack a network. This is more so in ad hoc networks, where the wireless interface and MAC protocol make the node more prone to attacks. Real network traffic is also not perfect since legitimate traffic often contains the kinds of patterns typically associated with attacks, which can significantly increase the false positive rate. It is therefore essential to reduce the rate of false positives for any IDS to be effective. Since the intrusion state cannot be inferred directly by monitoring any specific parameters, we need to predict an attack based on a 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 mixture component. Using this mechanism, an observed deviation from normal behavior carries a higher probability of being in a non-normal state (or one of the attack states). Given the computational complexity of the HMM models, it is not practical to execute them on all battery-limited host nodes. Therefore, we enhanced the model by distributing the HMM processing such that all nodes contribute to the HMM processing in a periodic manner. We also contributed to the concept of a feedback control mechanism that regulates the defensive response to every perceived abnormality. As explained earlier, this helps reduce the false positive rate, which is one of the major problems in modern IDSs. Modern silicon (CPU, I/O hubs, PCI express devices) contains performance counters that can be measured at moderate granularity. To avoid software overhead, these counters can be mapped to the feedback control modules. These modules can multiplex multiple measurements that help in battery and cost savings. While this methodology effectively solves some issues of IDSs, especially IDSs for MANET, it is not a complete solution; mechanisms that can provide more lead time in identifying early signs of attacker activities to minimize the damage are still needed.

Modern IDSs also lack automated response due to the high potential for inappropriate response and misdiagnosis. Damage recovery is another area for improvement, lack of which will make it difficult to create closed-loop control. High false positive rates is a serious issue because it can cause a situation where an administrator (or a user) can ignore all warnings. While a feedback control mechanism tackles this problem, future research in diagnostic accuracy are critically needed. Due to computational complexity, resource/battery limitations, and wireless-based vulnerabilities, we need IDS-friendly silicon. Various functional units of the silicon should be able to profile the activity trends supported by an eventing mechanism in a power-efficient manner. Physical layer design should support protocols related to optimal monitor node selection using user-defined policies and authentication. Additionally, industry needs to define advanced techniques related to node fingerprinting that profiles a remote system based on factors like received signal characteristics. This, in cooperation with the method defined in this article, can reduce false positives to a greater extent by profiling physical deployment of individuals to identify the attacker. Furthermore, it is no myth that the future of IDSs lies in data correlation that produces results by observing several different sources in a short timeframe. IDSs need to understand relationships, relevance, and correlation between multiple triggers (or emissions) in a computationally efficient manner. Control-oriented distributed HMM is the first step toward that goal. Trust management is another research area needed for successful implementation of IDSs in MANETs for generation and distribution of mutual data.

References

[1] S. Ci *et al.*, "Self-Regulating Network Utilization in Mobile Ad-Hoc Wireless Networks," *IEEE Trans. Vehic. Tech.*, vol. 55, no. 4, July 2006, pp. 1302–10.
[2] X. Du *et al.*, "A Secure Routing Protocol for Heterogeneous Sensor Networks," *Proc. IEEE GLOBECOM '06*, San Francisco, CA, Nov. 2006.

[3] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, Feb. 1989, pp. 257–86.
[4] R. Khanna and H. Liu, "System Approach to Intrusion Detection Using Hidden Markov Model," *Proc. 2006 Int'l. Conf. Commun. and Mobile Comp.*, July 2006, pp. 349–54.
[5] S. S. Joshi and V. V. Phoha, "Investigating Hidden Markov Models Capabilities in Anomaly Detection," *Proc. 43rd Annual Southeast Regional Conf.)*, Kennesaw, GA, Mar. 2005, pp. 98–103.
[6] A. Arnes *et al.*, "Using Hidden Markov Models to Evaluate the Risks of Intrusions: System Architecture and Model Validation," *Proc. Int'l. Symp. Recent Advances in Intrusion Detection*, Hamburg, Germany, Sept. 2006.
[7] D. Ourston *et al.*, "Applications of hidden Markov Models to Detecting Multi-stage Network Attacks," *Proc. 36th Annual Hawaii Int'l. Conf. (Sys. Sci. '03)*, Jan. 2003.
[8] W. Wang, X. -H. Guan, and X.-L. Zhang, "Modeling Program Behaviors by Hidden Markov Models for Intrusion Detection," *Proc. Int'l. Conf. Machine Learning and Cybernetics*, 2004, Aug. 2004, pp. 2830–35.
[9] S. Zanero, "Behavioral Intrusion Detection," *ISCIS '04*, 2004.
[10] D. Wagner and D. Dean, "Intrusion Detection Via Static Analysis," *Proc. IEEE Symp. Research in Sec. and Privacy*, Oakland, CA, 2001.
[11] S. Manganaris *et al.*, "A Data Mining Analysis of RTID Alarms," *2nd Int'l. Wksp. Recent Advances in Intrusion Detection*, Purdue Univ., West Lafayette, IN, Sept. 1999.
[12] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drifting and Hidden Contexts," *Machine Learning*, vol. 23, 1996, pp. 69–101.
[13] W. Fan, "Systematic Data Selection to Mine Concept-Drifting Data Streams," *ACM SIGKDD*, 2004.
[14] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Annals of Mathematical Statistics*, vol. 22, Mar. 1951, pp. 79–86.
[15] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, 2nd ed., Clarendon Press, 1992.
[16] T. K. Moon, "The Expectation-Maximization Algorithm," *IEEE Signal Processing*, Nov. 1996, pp. 47–59.
[17] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models," *IEEE Symp. Sec. and Privacy*, Oakland, CA, 1999, pp. 133–45.

BIOGRAPHIES

RAHUL KHANNA (rahul.khanna@intel.com) received his M.S. and P.D. degrees from Columbia University, New York, and is currently a Ph.D. candidate at Oregon State University. He is working at Intel as a senior architect in server manageability, power optimization, failure prediction and analysis, HS interconnects, and wireless technologies. He holds 10 patents in server technologies with many pending. He has authored peer reviewed papers in areas related to IBIST, imaging, sensor networks, and mobile/wireless security.

HUAPING LIU (hliu@eecs.oregonstate.edu) received his B.S. and M.S. degrees from Nanjing University of Posts and Telecommunications, China, in 1987 and 1990, respectively, and his Ph.D. degree from New Jersey Institute of Technology, Newark, in 1997, all in electrical engineering. From July 1997 to August 2001 he was with Lucent Technologies, New Jersey. He joined the School of Electrical Engineering and Computer Science at Oregon State University in September 2001, and is currently an associate professor. His research interests include ultrawideband and OFDM techniques, MIMO systems, mobile networks, and network security.