Finding The Needle: Suppression of False Alarms in Large Intrusion Detection Data Sets

James J. Treinen, Ramakrishna Thurimella
Colorado Research Institute for Security and Privacy
University of Denver
Denver, USA
jamestr@cs.du.edu, ramki@cs.du.edu

***Abstract*—Managed security service providers (MSSPs) must manage and monitor thousands of intrusion detection sensors. The sensors often vary by manufacturer and software version, making the problem of creating generalized tools to separate true attacks from false positives particularly difficult. Often times it is useful from an operations perspective to know if a particular sensor is acting out of character. We propose a solution to this problem using anomaly detection techniques over the set of alarms produced by the sensors. Similar to the manner in which an anomaly based sensor detects deviations from normal user or system behavior, we establish the baseline behavior of a sensor and detect deviations from this baseline. We show that departures from this profile by a sensor have a high probability of being artifacts of genuine attacks. We evaluate a set of time-based Markovian heuristics against a simple compression algorithm and show that we are able to detect the existence of all attacks which were manually identified by security personnel, drastically reduce the number of false positives, and identify attacks which were overlooked during manual evaluation.**

***Keywords*-intrusion detection; anomaly detection; markov chain; hidden markov model**

I. Introduction

The number of high profile network compromises continues to grow. Increasingly, the trend is toward profitable cyber crime, as opposed to the relatively benign Denial of Service (DoS) and worm attacks of the last decade. More and more, cyber criminals are organizing and turning their attention to corporate networks with the hopes of carrying out attacks resulting in the theft of digital information of real value. As a means of addressing this problem, we evaluate a set of automated techniques to assist the security staff of large corporate networks in the detection of malicious activity.

Generally speaking, the intrusion detection architectures for large networks are comprised of a set of host or network based intrusion detection sensors (IDSs) which monitor for malicious activity. There are two main types of intrusion detection systems: misuse and anomaly detection. Misuse detection systems use pattern matching techniques to detect malicious activity using a set of pre-defined signatures. The main drawback of this approach is that it requires that the IDS have pre-existing knowledge of an attack profile in order to detect instances of that attack. While misuse detection systems generally produce fewer false positives than anomaly detection systems, they do not have the ability to detect attacks which are not defined in their signature database. On the other hand, anomaly detection systems define a baseline profile for a system which is being monitored and raise alarms when significant deviations from this profile are detected. Anomaly detection systems are more flexible in detecting emerging attacks, but often exhibit higher false positive rates. It is common in enterprise computing environments to deploy a combination of both classes of sensors and perform analysis of the alarms generated by both systems at a central Security Operations Center, or SOC. The techniques described in this paper fall into the class of anomaly detection algorithms.

During the course of our experiments, we evaluated the performance of a set of alarm evaluation heuristics comprised of single step Markov Chains, Hidden Markov Models, and a simple heuristic based on the GNU gzip utility. The underlying intuition to our approach is that intrusion detection sensors are inherently noisy, and although the false positives they generate appear random, the behavior of a given sensor will exhibit a “normal” behavior which can be modeled over time. We further hypothesize that deviations from this “normal” behavior have a high probability of being attacks. It is important to note that a model must be created for each sensor as the software versions, signature databases, and placement of the device can vary significantly across the installation base. As such, no general model can be created to cover the set of sensors for the entire network. A potential weakness of this approach is the likelihood that the alarms generated by a particular sensor will vary over time, especially in the case of a major software update to the device. Events of this nature will require retraining of the models.

To support our hypotheses, we adapt earlier work from the field of applied statistics. Schonlau, et al. evaluate the efficacy of five statistical heuristics in detecting masqueraders via the statistical analysis of system call traces [23]. We adapt their approach to the analysis of IDS alarms, and

Research supported by NSF Grant DUE–0416969

show that Markov Chains and Hidden Markov Models prove to be very effective at detecting all types of attacks by acting as an anomaly detector over the set of IDS alarms. We also evaluate the compression technique described in [23] and show that while it is effective at intrusion detection, it yields a significantly higher percentage of false positives for this type of analysis. We do not evaluate the "Uniqueness" approach described by them because we do not perform our analysis on a per user or per IP address basis. Neither do we evaluate IPAM or the "Sequence Match" methods described in their paper, for similar reasons.

We chose to evaluate the alarm sequences on a per sensor basis as opposed to a per IP address basis. The IP space on any given network is extremely large, and analysis of the alarms generated per IP not only has the potential to require the modeling of millions of distinct IPs, but the number of alarms generated per IP address is not significantly large to lend itself to training a model of any kind. In addition to this fact, we chose to model at the sensor level of aggregation to fix a potential weakness in Ourston's work [18]–[20]. While Ourston, et al. made a significant contribution to the field of intrusion detection by introducing the concept of Markovian modeling to the field of IDS alarm analysis, it is ineffective when an attacker spoofs his IP address using a Sybil attack [8]. By not relying on connection records as our level of aggregation for alarm analysis, but rather aggregating one level higher at the sensor, our techniques are immune to this type of attack.

The remainder of this paper is organized as follows. Section II discusses related work. Section III provides an in depth discussion of the nuances of the data used in our experiments, and discusses experimental design and notation. Section IV formally defines our models, and provides results from each approach. An overview of the strengths and weaknesses of each technique and a discussion of open problems are provided in Section V.

II. Related Work

It is well established that it is possible to detect attacks based on deviations from normal system behavior by modeling a baseline set of system calls, and detecting anomalous activities via departure from this profile. We expand this concept to the evaluation of alarms and show that it is very effective at detecting attacks at the sensor level.

Little research has been performed in the area of profiling IDS sensors. Previous work consists mainly of research which was performed by the IBM Zurich Research Lab on techniques for creating sensor profiles using Association Rules [1]. Our approach differs from this work in that we take into account the order in which the alarms are generated by the sensor.

Krügel et al. propose a system for improving the accuracy of anomaly based IDS sensors using Bayesian networks [17]. Our approach differs from this in that Bayesian networks do not model the inter-dependence of alarms using time based sequencing. We show that there is a strong relationship between the order in which alarms are generated, and whether they in fact are the result of a genuine attack.

In the field of intrusion detection, Hidden Markov Models have been applied in various ways to the problem of learning normal user or process behavior based on system call traces in Unix. These generally have extended earlier work on modeling traces of Unix system calls using N-grams [13], or Markov Chains [14]. The application of Hidden Markov Models to Unix system calls generated by operating system processes is explored in [9], [27]. Ju presents research on using HMMs to model user generated system calls in [15]. The application of HMMs to network data is explored in [26], and [2].

The application of Hidden Markov Models to intrusion detection has received renewed attention after work by Ourston, et al. [18], [20] which presents a technique for detecting multi-stage attacks. The main weakness with this approach is its reliance on connection records, which are trivial to compromise if an attack originates from multiple source IPs or if an attacker spoofs their IP address using a Sybil Attack [8]. A secondary limitation of this approach is that they train the HMMs for positive response. If a new category of attack emerges, until a new HMM is trained for that attack, the alarm sequences falling into this category have a high probability of going undetected. Haslum et al. present a technique for quantifying risk to a network based on the set of alarms from multiple intrusion detection sensors in [12] and use this merged alarm stream to calculate a risk score using a Hidden Markov Model. They extend their work in [10], [11] to build an intrusion prevention sensor which predicts whether an alarm sequence has a high probability of being followed by an alarm which will complete an attack scenario and takes preventative action to mitigate the threat.

A general framework for the application of Markov Chains in anomaly based intrusion detection systems is given by Jha in [14]. A series of experiments is conducted using system call data. Jha's framework is frequently adapted by other researchers. Sallhammar presents a method for applying Markov Chains in conjunction with a cost and reward system for computing the probability of an attack based on game theory [22]. Khanna presents a novel approach for detecting attacks on mobile ad-hoc networks using Hidden Markov Models in [16]. Zanero uses a combination of Hidden Markov Models based on system call traces, and theory from the field of Ethology to create formalized characterizations of system interactions resulting in what he calls a "Behavioral Intrusion Detection System" [28].

The primary difference between our work and that of the prior art is that we build profiles which are intended to model normal behavior for a particular sensor. The majority of the prior research models specific attacks, or attack profiles, and attempts to make predictions based on those models [11].

In contrast, we model the baseline false positive noise of a sensor as "normal" behavior. By detecting deviations from this baseline, we are able to detect a change in the sensor state. We then show that this anomalous behavior has a high probability of representing malicious activity on the network.

III. On Data and Experimental Design

Using supervised training techniques as described in [18], [20] is extremely difficult. Significant portions of these two papers are dedicated to preprocessing routines which generalize the base alarm data to a form where creating abstract models of attacks is feasible. Our approach is fundamentally different. Rather than attempt to train our models to behave as misuse detection systems with the ability to detect predefined categories of attacks, we train our models to be anomaly detection systems. We establish baseline sensor profiles and use our models to detect deviations from the normal stream of false alarms which are emitted from by sensor.

Because we have access to a large repository of known security incidents, we generate training sequences of observations based on periods of data which contain no known attacks. This approach carries the inherent risk that the SOC personnel overlooked an attack that may be present in a training sequence. We mitigate this risk as much as possible by using a large number of training sequences, and closely examining false positives that are detected on the training data to ensure that no attacks were inadvertently introduced during the training period.

We conducted our experiments on the set of alarms generated by two IDS sensors running in production mode on large corporate networks. We label these sensors sensor A and sensor B. Sensor A was a Cisco NetRanger network IDS. Sensor B was a SourceFire network IDS Sensor. We selected these two sensors for our experiments to demonstrate that our techniques were technology agnostic, and to compare our results on sensors which had received differing levels of filtering and tuning. We also wished to conduct our experiments on sensors which are representative of typical technologies in use in current corporate environments. We evaluated the set of alarms generated by each sensor for the 30-day period starting May 1, 2008 and ending May 30, 2008. Sensor A was tuned to be relatively quiet, and generated 3483 alerts for this time period. Sensor B was not tuned as aggressively and monitored a larger network. Sensor B generated 172,839 alerts during the test period. During the test period, 17 of the 3483 alarms generated by sensor A were reflective of true attacks. For sensor B, 308 alarms represented genuine attacks. For both sensors, the false positive rate was well over 99%, making the discovery of genuine attacks extremely difficult.

Table I shows a typical set of IDS alarms consisting of the IP address of the attacker, the IP address of the victim, the numeric signature ID, and the name of the signature for which the alarm was raised. The only column of significance for our analysis is Signature ID. In order to analyze the alarms we map the Signature ID field from the alarm to an identity field in a database which we custom built to facilitate this analysis. The mapped signature IDs are monotonically increasing integers, ranging from 1 to m where m represents the number of distinct signatures, i.e. the number of different attack types, for which an alarm was raised during the 30-day test period. m = 22 for sensor A and m = 800 for sensor B.

Figure 1 is a plot of the signature frequency distribution for sensor A over 30 days. Figure 2 is the same plot with alarm frequencies for sensor B. In both cases, the

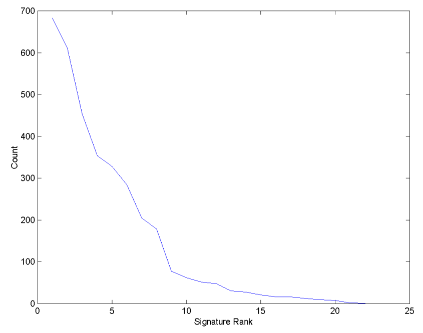

Figure 1. Sensor A Signature Distribution

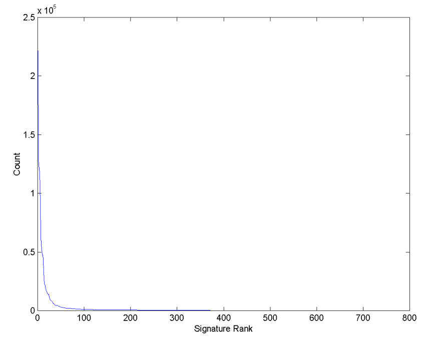

Figure 2. Sensor B Signature Distribution

Source IP	**Target IP**	**Signature ID**	**Signature**
10.0.0.1	10.0.0.4	**1**	TCP port scan
10.0.0.2	10.0.0.4	**1**	TCP port scan
10.0.0.3	10.0.0.4	**2**	sendmail overflow
10.0.0.5	10.0.0.7	**3**	ftp brute force login

Table I
Typical Intrusion Detection Alarms

vast majority of the alarm traffic for the 30-day period is comprised of a relatively small number of signatures, and drops off quickly for the remainder of the signature set. We have evaluated many other production sensors and found this to be typical for any given IDS. Given this phenomenon, it is normal to see that each of the sequences of alarms that we test look very similar in composition. This is the main inspiration for our research. Given that the majority of the alarms generated by an IDS are the same signatures over and over, it makes sense that deviations from these alarms, and the order in which they appear, are indicative in a change in state of the sensor, i.e. from emitting false positives, to detecting genuinely malicious activity on the network. This change in state is an artifact of a change in the type of traffic which is being monitored, i.e. from legitimate traffic that the sensor mis-classifies as an attack, to genuine attacks resulting in genuine alarms.

It is important to note that the set of alarms present in the training data, and those in the test data, are not mutually exclusive. If this were to be the case, separating legitimate alarms from false alarms would be the trivial exercise of simply filtering the "noisy" signatures. In fact, all signatures from the test data were represented in the training data as well. This further demonstrates that the order in which the alarms are generated is a significant indication of whether the alarms are false positives, or manifestations of an attack.

In order to build the data sets that were used in our experiments, we constructed training and test sequences in the following manner. Let $A = \{a_1, a_2, \ldots, a_n\}$ be the complete set of alarms generated by the sensor over the 30-day experimental time period. We subdivided A into two subsets consisting of training data R and test data S. We selected a period of days during which no known attacks were identified by the SOC and generated a set of training sequences R such that each $r_t \in R$ is a sub-sequence $\{r_t \ldots r_{t+k}\}$ beginning at time t. The set of sequences was generated using a sliding window of length k. For both sensor A and sensor B, this training data was comprised of the first 5 days of the month. We defined the set of test data as $S = \{A - R\}$ and generated test sequences $s_t \in S$ in the same manner as the training sequences. Originally we attempted to model the sensors in a state of silence by inserting a signature id of "0" for each second of the day during which no alarm was generated. Given that there are 86,400 seconds in a day, this had the effect of diluting the signal produced by the sensors to the point where analyzing the signal produced by the sensor became ineffective. As such, we made the decision to model only the actual signal, and not introduce the notion of silence to the models. On average sensor A generated an alarm every 12 minutes, and sensor B generated an alarm every 17 seconds. This is the same approach used in [18]–[20], [23]. A good topic for future research would be to introduce continuous time Markov Chains to the set of experiments, and model the absence or presence of alarms as a Poisson process. This would provide a facility for analyzing bursty behavior by a sensor, or a normally noisy sensor which suddenly goes quiet, both of which are potential indicators of malicious activity on either the network, or the sensor itself.

A. Experimental Design

We evaluated three different methods during the course of experiments, "Compression", "Single Step Markov Chain", and "Hidden Markov Model". All three of these methods attempt to detect anomalies in a stream of alarms generated by production intrusion detection sensors as a means of detecting attacks based on deviations from normal sensor behavior.

The methods share a common data foundation, in that the set of alarms is segmented into training data and test data, each of which are further subdivided into training and test alarm sequences. To facilitate discussion for the remainder of this paper, we define the following notation:

$A = \{R \cup S\}$	The set of alarms
R	The training data
r_t	The training sequence starting at time t
S	The test data
s_t	The test sequence starting at time t
M	The set of distinct integer signature IDS
m	The size of M
k	The length of the training and test sequences

The sequence length evaluated during all experiments was 10. This choice appears somewhat arbitrary, but it was determined during the course of the experiments that a sequence length of 10 yielded the best results. We also evaluated sequence lengths of 2, 5, 15, 25, and 50, all of which yielded inferior results for both sensors.

IV. Overview of Methods and Model Construction

A. Compression

1) Intuition: The underlying intuition behind the compression method [23] is that test data which are appended to a training data set will yield a higher compression ratio if they are similar to the training data than if they vary significantly. This is due to the nature of the compression algorithm used in the gzip utility, as defined in [25]. The underlying Lempel-Ziv algorithm builds compression rules starting from the beginning of the file to be compressed. Given this fact, it makes sense that as these rules are built from the front of the file, data appended to the end of the file will compress more readily if it is similar in nature to the data which was used to build the rules. If the appended test data differs significantly from the training data, the compression ratio will suffer. Informally, this method tries to capture changes in entropy [24] as test data is appended to the training data.

2) Approach: To score this approach we define a score x_{s_t} for each test sequence $s_t \in S$ as the number of additional bytes required to compress the test sequences when appended to the training data R

$$x_{s_t} = gzip(R + s_t) - gzip(R)$$

3) Thresholds: The threshold used in the compression experiments was determined by calculating a set of cross validated scores x_t^{cv} for each sequence in the training data for both sensor A and sensor B. For each training sequence $r_t \in R$ we compute

$$x_t^{cv} = gzip(R + r_t) - gzip(R).$$

We fixed our target detection rate at 100% for known attacks and experimentally determined the appropriate threshold for each of the two sensors. The resulting thresholds were the 97th percentile for Sensor A, and the 89th percentile for Sensor B. When evaluating the test sequences $s_t \in S$, any sequence receiving a score $x_{s_t} > threshold$, was marked anomalous.

4) Results: The compression algorithm, when applied to sensor A, generated 1021 meta-alarms, yielding an alarm reduction rate of 71%. When applied to sensor B, 19231 meta-alarms were generated, yielding an alarm reduction rate of 88%. Overall, the use of the gzip utility yielded the worst results of the three techniques explored in this paper. Rather than relying on the gzip utility to perform the calculations, a formal investigation of the efficacy of entropy based anomaly detection on IDS alarms may yield better results, and warrants further exploration.

B. Markov Chains

1) Motivation: Markov Chains and Hidden Markov Models come from the field of signal processing, and have been used extensively in various speech recognition and machine learning applications. The benefit of these two techniques lies in the fact that they model the the order in which events occur in a training data set, and can be used to evaluate the probability of a sequence of events from a test data set. It is intuitive that the order in which alarms occur is important in the detection of attacks, and that this order will differ from the order in which alarms are generated as false positives. We show that detecting these changes is a very effective means of detecting attacks in a network, with a low rate of false positives, and a high rate of alarm reduction.

2) Model: Markov Chains are stochastic processes which are effective at modeling the behavior of a system over time. A complete discussion of Markov Chains is provided in [7], which defines a Markov Process as

$$\left\{ X^{(n)}, n = 0, 1, 2, \dots \right\}. \tag{1}$$

which takes a finite or countable set M, in this case the integer signature ids emitted by IDS sensors.

As in the compression technique, we define A as the total set of alarms emitted over the 30-day experimental time period. We further divided A into two subsets $R \subseteq A$, the attack free training data, and $S \subseteq A$, the test data. R and S are decomposed into sub-sequences using the same sliding window technique described for the compression experiments such that $r_t \in R$ is the training sequence starting at time t and $s_t \in S$ is the test sequence starting at time t. s_t and r_t are of the same predetermined length k. As such,

$$M = \{0, 1, 2, 3, 4, \dots, m\}$$

which may be realized as:

$$\begin{aligned} s_t \;\; &= \{5, 7, 5, 6, 6, 6, 2, 4, 7, 7\} \\ s_{t+1} &= \{7, 5, 6, 6, 6, 2, 4, 7, 7, 3\} \\ s_{t+2} &= \{5, 6, 6, 6, 2, 4, 7, 7, 3, 2\} \\ s_{t+3} &= \{6, 6, 6, 2, 4, 7, 7, 3, 2, 9\} \\ &\vdots \end{aligned}$$

Definition 1. Suppose a fixed probability P_{ij} independent of time exists such that

$$P(X^{(n+1)} = j | X^{(n)} = i, X^{(n-1)} = j_{n-1}, \dots, x^{(0)} = j_0) = P_{ij}, \quad n \geq 0$$

where $\{j, i, j_0, j_1, \dots, j_{n-1}\} \in M$. Then this is called a Markov Chain process.

This probability can be interpreted as the conditional distribution of any future state $X^{(n+1)}$ given the past states

$$X^{(0)}, \quad X^{(2)}, \quad \dots, \quad X^{(n-1)}$$

and present state $X^{(n)}$ is independent of the past states and depends solely on the present state. The probability P_{ij} thus represents the probability that the process will transition to state j given that it is currently in state i.

The transition probability P_{ij} is contained in a transition matrix, which holds the transition probabilities between all states in the Markov Chain.

$$P = \begin{pmatrix} P_{00} & P_{01} & \cdots \\ P_{10} & \ddots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

We use the technique of maximum likelihood to fit our data to the Markov Chain Model, and estimate the values of P. P is known as the one-step transition Matrix, and holds the probabilities of transition from one state to another state in a single step.

In order to determine the probability of being in a certain state n steps from now, we must calculate the n-step transition matrix. We call these probabilities the outlook probabilities. Using the transition values from P and an initial probability vector X, we are then able to calculate an "outlook" probability as follows

Definition 2. Let $X^{(n+1)} = PX^{(n)}$ be the probability distribution of the states one step from time n. We then know that $X^{(n+1)} = P^{(n+1)}X^{(0)}$ and $X^{(n+1)}$ holds the probabilities of being in a given state at time $n+1$, given the initial probability distribution $X^{(0)}$ and the one-step transition matrix P.

We are then able to determine the probability of a sensor emitting an alarm n steps from the current time (t).

Definition 3. Let X, the initial probability distribution vector be constructed in such a way that given a sequence of alarms $s_t \in S$ beginning with the signature id s_{t_0}, let $x_{s_{t_0}} \in X^{(0)} = 1$ and all other $x \in X^{(0)} = 0$ indicating that the known starting state of the test sequence is s_{t_0} with probability 1. Given the one-step transition matrix P, we define the alarm outlook measurement to be

$$O_{s_t} = \prod_{n=1}^{k} P^{(n)} X^{(0)} \tag{2}$$

where t is the time of the first alarm in the sequence being evaluated, n is the n^{th} element of the sequence, and $k = 10$ is the sequence length, as before.

3) Threshold: The threshold for the set of experiments using Markov Chains was calculated in the following manner. Given the one step transition matrix P, and an initial state probability vector $X^{(0)}$, for each training sequence $r_t \in R$ calculate $\{P(r_t)|P, X^{(0)}\}$ using equation (2). We calculated the 99.9th percentile of these scores, sorted highest to lowest, and marked any sequence as anomalous which had a probability lower than the threshold determined by the training data.

4) Results: For this set of experiments we were able to detect 100% of those attacks which were manually identified by the SOC using $k = 10$ as the sequence length for both sensors. In addition to accomplishing the automation of attack detection in the alarm logs, we were able to successfully identify multiple attacks and reconnaissance events which had gone unnoticed during manual inspection of the alarms. Over the 30-day period, the Markov Chain anomaly detector raised 482 meta-alarms for sensor A, yielding a suppression rate of 86%. For sensor B 1230 meta-alarms were generated, yielding a suppression rate of 99%.

C. Hidden Markov Models

1) Model: Hidden Markov Models were first proposed by Baum in [3]–[6]. A Hidden Markov Model (HMM) is a doubly embedded stochastic process which models a set of symbol observations. Hidden Markov Models differ from basic Markov models in that the state which emits the observation is invisible, i.e. *hidden* from the observer. In a standard Markov process, the states themselves are visible to the observer. The observations in Hidden Markov Models are dependent on observation probability distributions at each hidden state, and transitions between the hidden states are governed by a secondary, hidden, stochastic process.

Rather than use the simple Markov Model described in the previous section, where each observation corresponds to a single state, the Hidden Markov model allows increased flexibility by modeling a set of observations as a probabilistic function of the current state, followed by a state change to either a new state, or the ability to remain in the current state prior to emitting the next observation, based on a state transition probability distribution. An in depth tutorial on Hidden Markov Models is presented by Rabiner in [21].

A Hidden Markov Model is defined by the following.

1) N. Let N denote the number of physical, hidden states of the model. This number is significant to some reality of state change in the real world which is represented in the model. After experimenting with different values for N, we found that $N = 2$ consistently produced the highest detection rates with he lowest false positive rates.
2) m. Let m denote the number of distinct observations that can be emmited per state. m is thus the size of the alphabet M of symbols which are actually observed by the user of the system. For the sensor profiling problem, m is the number of distinct IDS alert signatures which are produced by the sensor. The observation symbols are given as $V = v_1, v_2, \ldots, v_m$.
3) α_{ij}. Let α_{ij} denote the transition probability distribution for the hidden states such that
$$\alpha_{ij} = P[q_{t+1} = H_j | q_t = H_i], \quad 1 \leq i, j \leq N. \tag{3}$$
4) $B = b_j(l)$ Let $B = b_j(l)$ denote the observation symbol probability distribution in a given state j such that
$$b_j(l) = P[v_l \ at \ t | q_t = H_j], \ \ 1 \leq j \leq N, \ \ 1 \leq l \leq M. \tag{4}$$
5) Let $\pi = \pi_i$ denote the initial state probability distribution such that
$$\pi_i = P[q_1 = H_i], 1 \leq i \leq N. \tag{5}$$

Given this set of parameters a Hidden Markov Model can be fully specified as $\lambda = (N, m, \alpha, B, \pi)$.

As in the Markov Chain experiments, the set of IDS alarms, A is divided into two sets of sequences R and S where $r_t \in R$ and $s_t \in S$ represent the sequence of length k at time t. The parameters α, B, π are all estimated using the Baum-Welch algorithm using the set of training sequences $r_t \in R$ [21]. We train the HMM using 5 days of IDS alarms for which no attacks are known to have occurred. Once trained, we are able to determine the probability score of a test sequence $s_t \in S$, the probability of a sequence of alarms, using the Forward Verterbi Algorithm [21] as

$$x_{s_t} = Verterbi(s_t) \tag{6}$$

2) Thresholds: To determine the threshold for marking test sequences as anomalous we calculated the score $x_{r_t} = Verterbi(R_t)$ for each sequence r_t in the training data R. As in the previous experiments, we fixed our target detection rate at 100% of known attacks and adjusted the threshold to achieve this goal. In order to detect 100% of known attacks we set the threshold for Sensor A at the 99.7th percentile. For Sensor B, we were able to tighten this threshold to the 99.9th percentile and still achieve total attack detection. Any sequence from the test data s_t was marked as anomalous if $x_{s_t} < threshold$.

3) Results: Over the 30-day experimental time period, 239 meta-alarms were created for sensor A using the Hidden Markov Model approach, yielding an alarm suppression rate of 93%. For the same time period, 7813 meta-alarms were generated, yielding a 95% alarm reduction rate. As with the Markov Chain approach, we were able to detect attacks and reconnaissance activity which had gone unnoticed by the SOC.

V. Conclusions

Technique	Sensor	Reduction	Threshold
Compression	A	71%	97
Compression	B	88%	89
Markov Chain	A	86%	99.9
Markov Chain	B	99%	99.9
Hidden Markov Model	A	93%	99.7
Hidden Markov Model	B	95%	99.9

Table II
Summary of results for profiling heuristics

Table II summarizes the results of our experiments. In order to provide a stable point over which to compare the performance of the three profiling heuristics, we set the threshold for each technique at a value where we were able to detect 100% of the known attacks from the test data. As expected, Markov Chains and Hidden Markov Models out performed the use of compression to detect anomalies in sensor behavior. Markov Chains suppressed the greatest percentage of false alarms for the noisy sensor, Sensor B, eliminating 99% of the false positives. The use of Hidden Markov Models was more successful in eliminating false alarms on the quieter sensor, Sensor A, yielding a suppression rate of 93%.

The relatively small number of alarms produced by Sensor A, overall, made it more difficult to train the models. As such, the performance of the tested techniques for Sensor A is not as good as for Sensor B. This can be attributed to the smaller amount of training data, and the greater mean time between alarms emitted by this sensor.

It is interesting that the compression algorithm performed as well as it did, given the small sequence lengths which were evaluated. For example, compression missed only one attack comprised of a single alarm on a day where the other 4284 alarms were all false positives. By definition, these "one shot, one kill" attacks are extremely difficult to detect due to the small footprint they leave in the data. Because of this, it is not surprising that simple compression was not enough to detect the existence of such an attack. This attack was detected by both the Markov Chain and HMM techniques, solely because it represented an anomalous state transition in a sequence of alarms that would otherwise be representative of normal system behavior on the part of the sensor.

Overall, we were able to suppress a very large number of the false alarms which were generated by both sensors. This has the net effect of reducing the work load of SOC personnel, while increasing the accuracy of the monitoring infrastructure as a whole.

Interesting further research on this topic would involve exploration of the alarm rate produced by a sensor. It is intuitive that significant changes in the rate in which an IDS emits alarms could be indicative of attacks. The authors suggest exploring this problem in terms of Poisson processes and continuous time, multi-step Markov Chains.

References

[1] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 115–118, 1997.

[2] D. Ariu, G. Giacinto, and R. Perdisci. Sensing attacks in computers networks with hidden markov models. In *MLDM*, pages 449–463, 2007.

[3] L. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

[4] L. Baum and J. Egon. An inequality with applications to statistical estimation for probabilistic functions of a markov process. *Bulletin of the American Meteorological Society*, 73:360–363, 1967.

[5] L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat.*, 30:1554 – 1563, 1966.

[6] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat.*, 41:164–171, 1970.

[7] W. Ching and M. Ng. *Markov Chains Models, Algorithms, and Applications*. Springer Science+Business Media, Inc., New York, 2006.

[8] J. Douceur. The sybil attack. In *IPTPS*, pages 251–260, 2002.

[9] Y. Du, H. Wang, and Y. Pang. Hmms for anomaly intrusion detection. In *CIS*, pages 692–697, 2004.

[10] K. Haslum, A. Abraham, and S. Knapskog. Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In *IAS*, pages 183–190, 2007.

[11] K. Haslum, A. Abraham, and S. Knapskog. Fuzzy online risk assessment for distributed intrusion prediction and prevention systems. In *Tenth International Conference on Computer Modeling and Simulation, UKSIM*, pages 216–223, 2008.

[12] K. Haslum and A. Årnes. Multisensor real-time risk assessment using continuous-time hidden markov models. In *CIS*, pages 694–703, 2006.

[13] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.

[14] S. Jha, K. Tan, and R. Maxion. Markov chains, classifiers, and intrusion detection. In *CSFW*, pages 206–219, 2001.

[15] W. Ju and Y. Vardi. A hybrid high-order markov chain model for computer intrusion detection. *National Institute of Statistical Science Technical Report Number 92*, 1999.

[16] R. Khanna and H. Liu. Distributed and control theoretic approach to intrusion detection. In *IWCMC*, pages 115–120, 2007.

[17] C. Krügel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *ACSAC*, pages 14–23, 2003.

[18] D. Ourston, S. Matzner, W. Stump, and B. Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[19] D. Ourston, S. Matzner, W. Stump, and B. Hopkins. Applications of hidden markov models to detecting multi-stage network attacks. In *HICSS*, page 334, 2003.

[20] S. Ourston, S. Matzner, W. Stump, and B. Hopkins. Coordinated internet attacks: responding to attack complexity. *Journal of Computer Security*, 12(2):165–190, 2004.

[21] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[22] K. Sallhammar, B. Helvik, and S. Knapskog. On stochastic modeling for integrated security and dependability evaluation. *JNW*, 1(5):31–42, 2006.

[23] M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and V. Yehuda. Computer intrusion: Detecting masquerades. *Statistical Sciences*, 16(1):1–17, 2001.

[24] C. Shannon. A mathematical theory of communication. In *Bell System Technology Journal*, pages 379–423,623–656, 1948.

[25] T.A. Welch. A technique for high performance data compression. *IEEE Computer*, pages 8–18, 1984.

[26] Y. Yasami, M. Farahmand, and V. Zargari. An arp-based anomaly detection algorithm using hidden markov model in enterprise networks. In *ICSNC*, page 69, 2007.

[27] N. Ye, Y. Zhang, and C. Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.

[28] S. Zanero. Behavioral intrusion detection. In *ISCIS*, pages 657–666, 2004.