Projecting Cyber Attacks Through Variable Length Markov Models

Daniel S. Fava, Stephen R. Byers, Shanchieh J. Yang
Department of Computer Engineering
Rochester Institute of Technology

***Abstract*—Previous works in the area of network security have emphasized the creation of Intrusion Detection Systems (IDSs) to flag malicious network traffic and computer usage, and the development of algorithms to analyze IDS alerts. One possible byproduct of correlating raw IDS data are *attack tracks*, which consist of ordered collections of alerts belonging to a single multi-stage attack. This paper presents a Variable Length Markov Model (VLMM) that captures the sequential properties of attack tracks, allowing for the prediction of likely future actions on ongoing attacks. The proposed approach is able to adapt to newly observed attack sequences without requiring specific network information. Simulation results are presented to demonstrate the performance of VLMM predictors and their adaptiveness to new attack scenarios.**

I. Introduction

The growth of information in digital format and the pervasion of computer networks into human activity have amplified the importance of managing data access and the channels through which data flows. Information and cyber security are multifaceted and entail the provision of user accounts and passwords to protect data, the encryption of communication mediums, the imposition of network access rules through firewalls, etc. In addition to such preventive methods, Intrusion Detection Systems (IDSs) are often employed to monitor networks for traces of malicious actions and security breaches. IDSs work by performing anomaly detection or misuse detection on host computers or network traffic. Several IDS solutions have been proposed, and taxonomies have been provided [1], [2].

As the complexity and size of networks grew, the increasing number of deployed IDSs generated a number of alerts that became overwhelming to human analysts [3], [4]. Many methods for creating comprehensive alert reports have been proposed as potential solutions to this problem. Some of these efforts were in the area of alert aggregation [5], [6], alert correlation [7], [8], [9], [10], [11], current and future threat analysis [12], [13], and projection of likely future attack actions [14], [15]. Among these, threat projection is an important step towards the mitigation of critical attacks.

Previous works in the area of attack projection heavily rely on a priori knowledge of network and system configurations, and are, therefore, challenged by the diversity and the ever-changing nature of system settings and exploitation methods. Alternatively, the framework presented here models malicious network activity and extracts relevant information about an attacker's overall behavior and intent without requiring knowledge of the underlying network configuration. The attacker's probable future attack steps and behavior are inferred from a Variable Length Markov Model (VLMM) created from previously observed as well as ongoing attack sequences.

The method presented in this paper borrows from sequence modeling techniques that have successfully been implemented in fields such as text compression, computational biology (DNA sequence analysis), and data forecasting. By interpreting cyber attacks as sequences of malicious actions observed through IDS alerts, this research is able to investigate the application of sequence modeling techniques in the context of cyber threat projection.

II. Related work

A. Previous approaches

Qin and Lee were one of the first to propose a high level attack projection scheme [15]. They investigated the use of an alert correlation system that was also capable of performing attack prediction. It applied Bayesian networks to IDS alerts in order to identify attack sequences and to predict possible future attack steps. Its high level correlation was based on plan recognition and required the creation of several possible attack plans by domain experts. Challenging steps in such an approach are 1) the creation of attack plans that are general enough to capture a myriad of attack behavior, but specific enough to provide insights about an attacker's goals, and 2) the search for a match between an unfolding multistage attack among the many devised attack plans. Similar to Qin and Lee's work, Mehta *et al.* [16] also employs pre-constructed probabilistic attack graphs to rank attack tracks.

The problem of projecting possible future cyber attacks was also explored by Holsopple *et al.* [13]. In this work, three pieces of information were used in order to identify likely attack targets: network topology, a mapping between services and host computers, and observed attack sequences. Given this information, a threat score was assigned to different entities of the network. More recently, a similar work that assigns an overall risk score to entire networks as well as to individual hosts was presented by Arnes *et al.* [12].

Li *et al.* [14] proposed an approach to assess threat by anticipating likely attacks. The authors were able to identify short sequences of probable attack steps, and to construct attack graphs by employing data-mining techniques on a dataset containing representative attack actions.

The approach described in this paper is complementary to risk assessment algorithms presented by Arnes *et al.* [12] and Holsopple *et al.* [13]. Similar to Li *et al.* [14], likely sequences of attack actions are identified. However, the objective here is to study attack behavior and to predict an attacker's future actions. Unlike Qin and Lee [15] and Mehta *et al.* [16],

who relied on expert knowledge and the pre-construction of probabilistic attack graphs, the approach described here loosely employs domain specific information.

Since network and behavior modeling are two complex tasks, decoupling them will allow the selection of a specialized solution tailored to the problem at hand. The approach presented here decouples the task of network modeling from attack behavior modeling by employing universal predictors on high-level attack information. The end result is a model capable of predicting probable attack actions given an attacker's unfolding sequence of malicious activities.

B. Sequence modeling

Modeling can be performed based on event types and their frequency of occurrence, on the time properties of events (duration, time interval, etc.), or on the ordering within event sequences. In the cyber domain, these approaches have been used when performing intrusion detection. For example, it has been shown that statistical tests applied to TCP/IP packet fields and the interval of arrival between packets are indicative of Denial of Service (DoS) attacks [17], [18], [19]. Instead of looking at event frequencies or their time properties for intrusion detection, we model and project attack behavior by investigating the sequential properties of correlated IDS alerts; therefore, an overview of sequence modeling applied in the context of intrusion detection is presented next.

The analysis of program behavior through sequences of system calls is an important example of how sequence modeling techniques have been applied in the cyber security domain. As shown in [20], [21], [22], [23], [24] and [25], a program exploited, such as through a buffer overflow attack, will produce a sequence of system calls that often differs from the ones generated during the program's regular usage. Even though evading attack methods have been explored [26], the analysis of sequence of system calls is an important component when detecting vulnerability exploitations.

Sequence modeling has also been used when finding exploitations through the command line. Models for users' command line patterns are proposed in, *e.g.,* [27], [28]. Probabilities for newly observed commands derived based on a history of command sequences are used to determine whether an attacker has hijacked an authentic user account, or whether a disgruntled user is compromising the network. Similar approaches have been employed when finding anomalies in system log messages. Ye *et al.* [29] proposed to find anomalies by computing the probability of a sequence of log messages given the model generated under regular conditions.

Like the works described previously, the approach presented here focuses on the ordering of events and on sequence models such as Markov and Hidden Markov Models (HMMs). However, unlike approaches described previously, the proposed work is *not* an *intrusion detection* system. Instead, it builds upon existing IDSs and alert correlators to extract attack patterns and predict attack behavior from identified and correlated IDS alerts. In summary, this paper addresses the problem of projecting cyber attacks by looking at the sequential properties of correlated IDS alerts belonging to multi-stage attack tracks.

C. Prediction

One goal of the proposed sequence model is to predict probable future actions belonging to an attack track. The problem of prediction has been addressed from many fronts. For example, consider a long sequence of symbols $\{x_1, x_2..., x_m\}$. In 1992, Ehrenfeucht and Mycielski proposed a simple predictor that looked for the longest repeating suffixes of a sequence, which is called the *maximal suffix* [30]. Ehrenfeucht and Mycielski predicted x_{m+1} to be equivalent to the symbol following the most recent maximal suffix.

Jacquet *et al.* modified Ehrenfeucht's and Mycielski's predictor by searching for a fraction of the maximal suffix in the original sequence [31]. The fraction of the maximal suffix was set to a size $\lceil \alpha D_m \rceil$ such that the fractional suffix would occur more than twice in the original string. Each of such occurrences was called a marker. The algorithm then performs a majority vote among the symbols that immediately follow the markers. It was shown that this scheme yields to a universal predictor, which is an optimal predictor whose average error converges to the minimum regardless of the generating source, as long as the source is stationary [31]. Other universal predictors have also been proposed, including the seminar paper by Feder *et al.* [32].

This work investigates the use of several finite-context models (also known as Markov Models), including a Variable Length Markov Model (VLMM). Similar to the algorithm proposed by Jacquet *et al.* , finite-context based predictors also take into account the frequency count of previously observed symbols and their contexts. However, in addition to being able to extrapolate from these sequences when anticipating likely future attack actions, the model also allows for the calculation of $P_n(x)$, which is the probability of occurrence of x given that an n-order model has been created from several representative attack sequences. A detailed description of the approach and the proposed algorithm is given next.

III. Methodology

A. Attack track preprocessing

Attack tracks are one possible byproduct of running a correlation engine on raw IDS alerts. These tracks consist of ordered collections of alerts belonging to a single multi-stage attack. An XML representation of a partial attack sequence is shown in Figure 1. Each track is composed of several <Alert> tags and other sub-tags. As shown next, an attack behavior model is built based on information contained in these fields.

Let the collection of attack tracks be $\Psi = \{\sigma_i,\ i = 1, 2, 3...\}$, where each attack track σ_i is a time-stamped ordered set of intrusion alerts $\{a_{i,1}, a_{i,2}, a_{i,3}...\}$. Let $a_{i,j}$ be an intrusion alert composed of many fields $(v_1, v_2, v_3...)$. For example, v_1 may represent the type of exploitation method of $a_{i,j}$, while v_2 may represent the attack target IP, and so on.

$$\begin{aligned}\sigma_i &= \{\ a_{i,1}\ ,\ a_{i,2}\ ,\ a_{i,3}\ ,...\} \\ &= \left\{ \begin{pmatrix} v_1 \\ v_2 \\ \cdots \end{pmatrix}_{i,1}, \begin{pmatrix} v_1 \\ v_2 \\ \cdots \end{pmatrix}_{i,2}, \begin{pmatrix} v_1 \\ v_2 \\ \cdots \end{pmatrix}_{i,3}, ... \right\}\end{aligned}$$

```
<Track>
  <Alert>
    <Snort>
      <DateTime>
          2005-10-04 14:34:47.503911
       </DateTime>
      <Description>
          ICMP PING NMAP
      </Description>
      <Protocol>ICMP</Protocol>
      <Source_IP>192.168.222.4</Source_IP>
      <Dest_IP>100.20.0.0</Dest_IP>
      <GId>1</GId>
      <SId>469</SId>
      <SId_rev>3</SId_rev>
      <Classification>
          Attempted Information Leak
      </Classification>
      <Priority>2</Priority>
    </Snort<
    <Sensor_ID>snort232</Sensor_ID>
    <Category>Recon_Scanning</Category>
  </Alert>
  <Alert> ... </Alert>
</Track>
```

Fig. 1. A partial attack track example in XML.

Let $a_{i,j} \in \Omega$ where Ω is the space defined by the set of fields of an alert message. Note that the richness of the set of fields of an alert message depends on the types of IDSs deployed and on the correlation engine used to create attack tracks.

An important step in the proposed approach is to translate each attack track $\sigma_i \in \Psi$ into a sequence of symbols $s_i = \{x_{i,1}, x_{i,2}, x_{i,3}, ...\}$, where each symbol represents an attack action captured by an IDS alert. For example, symbol $x_{i,j}$ represents the j^{th} alert of track σ_i. Once attack tracks have been converted, different sequence modeling schemes, such as various Markov models, may be used to characterize their corresponding sequences of malicious actions.

When building the behavior model, not all IDS alert fields must be considered. Often, one is interested in a subspace of Ω. For example, if only the k^{th} alert field is relevant, then Ω_k is the space in which $a_{i,j} = (0, 0, ..., v_k, 0...)$. In this work, the alert fields that define Ω are shown in bold in Figure 1. Three different subspaces of Ω are considered. One subspace takes into account the attack description (Ω_d), another takes into account the category of the attack (Ω_c), and one accounts for the destination IP of the attack combined with the attack's category (Ω_t). These are derived from the <Description>, <Category> and <Dest_IP> tags of Figure 1.

Alert description contains a detailed account of the attack method, while the category field contains a coarser definition of the attack. For example, one possible category is an *Intrusion Root*. Within this category, there are WU-FTP exploitations, different WEB-IIS exploitations, and many others that may lead to an intruder gaining root access to a network computer. An initial analysis of the trade-offs in creating models from information at high and low levels of granularity (such as attack category versus description) was presented in [33].

Different choices of alphabets are applicable to the proposed VLMM approach; these choices may offer different insights into attacker's behavior. For example, description (Ω_d) and category (Ω_c) fields reflect the attacker's tendency in choosing different reconnaissance and exploitation methods and are used in this paper to analyze the effectiveness of modeling at two different levels of granularity. The field 'classification' is another choice that captures the attacker's method, but at a granularity between category and description. The incorporation of destination IP of attack steps (Ω_t) is also considered since it allows the model to account for the path and 'agility' of the attacker. Further research can also explore the benefits of employing additional subspaces of Ω such as destination port, protocol, and any combination of the fields.

```
<Track>
  <Alert>
    <Description>
        ICMP PING NMAP
    </Description>
    <Dest_IP>100.20.0.0</Dest_IP>
    <Category>
        Recon_Scanning
    </Category>
  </Alert>
  <Alert>
    <Description>
        SCAN SOCKS Proxy attempt
    </Description>
    <Dest_IP>100.10.0.1</Dest_IP>
    <Category>
        Recon_Scanning
    </Category>
  </Alert>
  <Alert>
    <Description>
        WEB-IIS nsiislog.dll access
    </Description>
    <Dest_IP>100.10.0.1</Dest_IP>
    <Category>
        Intrusion_Root
    </Category>
  </Alert>
</Track>
```

Fig. 2. Representing an attack track as different sequences of symbols.

Figure 2 shows the conversion of a single attack track into sequences of symbols. Each of the three alerts belonging to the track is mapped into a different symbol depending on the choice of Ω_k, $k \in \{c, d, t\}$. From the point of view of alert's category, the two first attack actions in Figure 2 were caused by reconnaissance scanning activities, while the third alert corresponds to an intrusion as root; therefore, this track is translated to the sequence of symbols AAB, where A corresponds to the category Recon_Scanning, and B represents Intrusion_Root. From the point of view of the alert's description, the first two alerts were "ICMP PING NMAP" and "SCAN SOCKS Proxy attempt," while the intrusion was an exploitation of the WEB-IIS server through the nsiislog dynamically linked library (DLL). Therefore, this track is translated to ABC, where A corresponds to "ICMP PING NMAP," B to "SCAN SOCKS Proxy attempt," and C to "WEB-IIS nsiislog.dll access."

Another important piece of information is the destination

of the attack steps, that is, the IP of the attack targets. When taking into account the attack target IP, this track is translated to AaB where A and a correspond to the category Recon_Scanning, with the lowercase letter representing a change in IP since the previous attack. That is, the transition in attack target between the first and the second attack steps of Figure 2 is represented as a lowercase character. The third attack step is represented as an uppercase B, because its destination IP is the same as the previous alert's.

Sequences created based on the three subspaces Ω_k, $k \in \{c, d, t\}$ are used to create three different models. These sequences of symbols are the input to the attack projection framework discussed in this paper.

B. Sequence modeling

The next task is to build a model that captures the ordering properties of sequence symbols. Finite-context models, typically known as Markov Models, and finite-state models, also known as Hidden Markov Models (HMMs), are two main approaches to sequence characterization.

Finite-context models assign a probability to a symbol based on the context in which the symbol appears [34]. In an n^{th} order finite-context model, an event is said to depend on n previous observations, whereby the transition probability is defined as $P_n\{X_{t+1} = x | X_{t-n+1} = x_{t-n+1}, ..., X_t = x_t\}$. On the other hand, *finite-state* models, also known as Hidden Markov Models (HMMs), are composed of an observable part (events), and a hidden part (states) [35]. Events are observed with different probability distribution depending on the state of the system.

For the results presented in this paper, attack tracks are modeled with finite-context models because they allow us to investigate how an attacker's current actions are correlated with recently observed ones. If IDS alerts belonging to a single attack track represent several attack steps performed by an intruder, then an n^{th} order finite-context model determines the probability of a future attack step based on the attacker's n previous actions. Looking back at the example of Figure 2, a finite-context model created based on attack categories (Ω_c) will help understand the relationship between root intrusions that are preceded by reconnaissance activities. Similarly, a model created from attack description (Ω_d) contains data at a finer level of granularity. In this case, the relationship between different pings, scans, and WEB-IIS exploitations, for examples, will be under analysis. In addition, incorporating destination IPs (Ω_t) allows the observation of how each attacker navigates through the network.

For example, consider the sequence:

$$+FGGFGF* \tag{1}$$

where $+$ and $*$ represent the start and the end of a sequence, and F and G correspond to Snort alerts 'WEB-IIS nsiislog.dll access' and 'WEB-MISC Invalid HTTP Version String' respectively. One may be interested in the probability of seeing symbol G after having observed F. This is captured by the first order Markov Model $P_1\{G|F\}$. In (1), F was followed by G in two out of three occurrences of F, and by $*$ once; therefore, a probability $P_1\{G|F\} = 2/3$ is obtained. In our work, *numerous* attack sequences will be taken into account when computing the transition probabilities for each of the n^{th} order models.

A first order Markov model can be represented by a squared transition probability matrix in which entry $p_{i,j}$ holds the probability of a transition from state i to j. A second order Markov model can be represented with a rectangular matrix in which $p_{ij,k}$ holds the probability of a transition into state k given that states i and j have been observed. Similarly, higher order Markov models can be represented in matrix forms.

Instead of using several matrices, one for each model order, this work implements different order Markov models using a *suffix tree*. The complexity for finding a context of size n and its succeeding symbol (x_{n+1}) in the suffix tree is $O(n)$. Yet, the suffix tree naturally holds all sub-contexts (suffixes) of a sequence. In other words, it holds all fixed order models in a single data structure. These models can then be combined into a Variable Length Markov Model (VLMM) [34]. For example, Figure 3 shows the suffix tree for the attack sequence shown in (1). Notice that all suffixes and their frequencies are captured by the tree: $F*$, GF*, $FGF*$, $GFGF*$, $GGFGF*$, $FGGFGF*$ and $+FGGFGF*$. Figure 4 shows the pseudo code for building a suffix tree from a set of attack sequences.

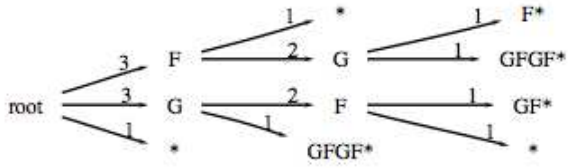

Fig. 3. Suffix tree for the finite sequence '$+FGGFGF*$'.

C. Prediction

Let $s = \{x_1, x_2, ..., x_m\}$ be an unfolding sequence of attack actions. Our objective is to predict the next action (x_{m+1}) given s and given a data-set containing representative attack tracks. Let $P_n(x)$ denote the probability of the next symbol in s being x according to an n^{th} order Markov model. Note that n can be at most m, which is the length of the already observed context. For a given set of representative attack sequences, the entire context of length m may not match any suffix in the tree; that is, this newly observed sequence s is not part of the data-set. Let H be a set containing all suffixes of representative attack tracks (in the case of this research, H is contained in a suffix tree), and l be the size of the longest suffix of s such that $s_l = \{x_{m-l+1}, ..., x_m\}$ exists in H. A VLMM determines $P(x)$ by combining or blending $P_i(x)$, $\forall\ i = \{-1, 0, 1, ..., l\}$ according to (2). The intuition is that past observations (captured by the symbols following suffixes of s that exist in H) are representative of the future.

$$P(x) = P\{X_{m+1} = x\} = \sum_{j=-1}^{l} w_j \times P_j(x) \tag{2}$$

The prediction process is illustrated in the pseudo code of Figure 5. The model is capable of generalizing by combining information belonging to previous observations that may be

```
learn_sequence( string sequence )
 for offset from 1 to sequence.length do
  current_node = root
  current_node.incident_edge_frequency += 1
  for i from offset to sequence.length do
   child = current_node.get_child( sequence[ i ] )
   if ( child != null )
    current_node = child
    current_node.incident_edge_frequency += 1
   else
    child = createBranch( current_node,
                          sequence, i,
                          sequence.length - i )
    break
   end if
  end for
 end for
```

```
create_branch( suffix_tree_node parent,
               string sequence,
               int offset, int length )
 descendent[ 0 ] = parent
 for i from 1 and j from offset, i < length + 1 do
  new_node = sequence[ i ]
  new_node.incident_edge_frequency = 1
  new_node.parent = descendent[ i -1 ]
  descendent[ i ] = new_node
 end for
 return descendent[ length ]
```

Fig. 4. Suffix tree creation pseudocode

```
predict( string context, character c )
 accumulated_escape = 1
 for i from 1 to context.length do
  sub_context = context[ i, context.length ]
  append_context = sub_context & c
  nom = # matches for append_context in suffix tree
  denom = # matches for sub_context in suffix tree
  probability = nom / denom
  escape_prob = compute_escape(context)
  prediction = prediction + probability × (1 - escape_prob )
                          × accumulated_escape
  accumulated_escape = accumulated_escape × escape_prob
 end for
 escape_prob = compute_escape(context)
 nom = number of matches for c in suffix tree
 denom = total number of observed characters
 probability = nom / denom
 prediction = prediction + probability × (1 - escape_prob )
                         × accumulated_escape
 accumulated_escape = accumulated_escape × escape_prob
 probability = 1 / alphabet_size
 prediction = prediction + probability × accumulated_escape
 return prediction
```

```
compute_escape( string context )
 nom = number of characters following context in suffix tree
 denom = number of matches for context in suffix tree
 escape_prob = 1 / ( 1 + denom )
 return escape_prob
```

Fig. 5. Prediction pseudocode

distinct from one another. The model uses a weighting scheme to combine these distinct observations contained in the different suffixes. The weights are computed in terms of escape probabilities as shown in (3) with $w_l = 1 - e_l$.

$$w_j \quad = \quad (1 - e_j) \times \prod_{k=j+1}^{l} e_k, \; -1 \leq j < l \qquad (3)$$

Escape probabilities work by allocating "some code space in each model to the possibility that a lower-order model should be used to predict the next character" [34]. There are different approaches to computing escape probabilities. An empirical test between the methods described in [34] showed no performance difference among them [33]. Therefore, a method that allocates one additional count over and above the number of times the context has been seen in each model order was implemented due to its simplicity. Specifically, $e_j = 1/(C_j + 1)$.

D. Realization of Proposed Methodology

The proposed methodology has three main components: alphabet creation and update (where sequences of symbols are created from attack tracks), sequence modeling, and prediction. Because of the computational simplicity — polynomial time for both sequence modeling and prediction — the entire system can be developed as an off-line or a real-time process. In the off-line mode, representative finite attack sequences are fed into the system to train the suffix tree model, which is then used to make predictions when live attacks are unfolding. In the real-time case, individual attack actions are appended to the model as live attack sequences are unfolding. As the suffix tree is built and updated with unfolding attack sequences, the model adapts to newly observed symbols and sequences. This adaptive ability, to be shown in Section IV-C, is critical since attacks and attack sequences could evolve due to changes in network configurations and discoveries of new vulnerabilities.

One implementation issue is that the real-time system will eventually run out of resources (memory) if it is indefinitely fed with new attack tracks. Future works will address questions on capacity, pruning of the model, and *minimum sufficient statistics*[1]. While the real-time implementation is more viable for its usefulness in the real world, the off-line development allows us to analyze the results in a more controlled setting, where VLMM predictions for different sequences and different symbols in the same sequence are made based on the same model trained.

IV. Experiments and Discussions

A. Experiment Setup

Both off-line and real-time implementations of the proposed approach were tested: one to analyze the VLMM's ability in capturing sequential orders of cyber attacks, and another to investigate the adaptiveness of the model to new attack actions in a real-time environment. In the first experiment, a data-set is randomly split into two halves. One half of the tracks are used to train the attack behavior model, and the other half to test the model's predictions. An important goal of this experiment is to examine the effectiveness of predicting by combining multiple fixed order Markov models. On the other hand, the second experiment utilizes a real-time implementation of the VLMM that processes alert messages one by one based on alert time stamp. This experiment will demonstrate the methodology's ability to adapt to new attack scenarios.

[1]Minimum sufficient statistics refers to the most compact summary of the data which retains all predictively-relevant information [36].

Note that the proposed methodology aims at projecting future actions for *each attack track*. Commonly known cyber security data-sets, including those from the MIT Lincoln Lab [37] [38], KDD Cup 99 [39] and Defcon [40], are crafted for intrusion detection and, thus, do not have the notion of attack tracks. More specifically, these data-sets are typically used to separate malicious activities from normal ones, but do not have the ground truth that specifies which malicious activities are executed as part of a multi-stage attack. The alert messages in some of the public data-sets may be sent through an alert correlator (such as those illustrated in Sections I and II) in order to produce correlated attack tracks. While this is desirable in a real-life system, the lack of multistage scenarios in these data-sets could introduce errors to the attack tracks, and, consequently, affect the confidence in the results obtained when analyzing the proposed attack projection system. In the absence of a proper public data-set for our experiments, this work utilizes one that is created for military applications and contains the ground truth of attack tracks.

The data-set used in this research was created through scripted multi-stage attacks performed on a VMware network[2]. Note that the proposed methodology does not require knowledge about specific network topology or configurations. As network configuration or attack strategy evolves, the proposed methodology shall adaptively adopt new symbols and new sequences of symbols in the model. In fact, the data-set contains different attack scenarios that target different parts of the network and utilize different exploitation methods. This allows for testing the proposed system's adaptiveness.

The network used for our reported results contains 7 internal subnets (each having a number of user address spaces), 22 external servers, and 24 internal servers. Example servers include IIS Web servers, MS Exchange Servers, FTP, and VPN servers that run on various Linux and Windows OS's. Five sets of attack scenarios are executed, producing a total of 19,908 alerts and 2,559 attack tracks. The attack scenarios range among CGI Overflow, Data Exfiltration, Phishing, and Deniel-of-Service, and differ in the attack targets. Alert messages were produced by Snort, Dragon, Apache, and IIS. A real-world operational system should have a data alignment pre-processing component that homogenizes alert messages produced by different types of IDSs [9], [11]. A homogenization step was not performed on the data-set used for this research. Instead, we used solely Snort alerts, which encompass approximately 50% of the total alerts and cover 55% of the attack tracks.

Additional study of the data reveals that some attack tracks contain a series of alert messages that are identical in every single field except for the time stamps and the IDS reporting the event. These messages could be duplicate alerts for the same attack action, or they could indeed correspond to a series of identical attack actions, which is not uncommon for scripted attacks. Regardless, consecutively repeated alerts add little value for a cyber attack projection system [33]. This work aims at extracting changes within attack tracks, and to predict symbols that are different from the latest observed one in the sequence. Therefore, repeated consecutive alerts are filtered, which helps reduce the model size [33].

Let $s_i = \{x_{i,1}, x_{i,2}, ..., x_{i,m}\}$ be a recently observed and unfolding sequence of attack actions captured by IDSs and translated from an attack track. A prediction set $\mathbb{P}$ composed of the most likely future steps is computed for $x_{i,m+1}$, for all sequences s_i in the test-set. The prediction performance is measured as the percentage of correctly predicted attacks over the total number of predictions. They are given in terms of top-1, top-2, and top-3 predictions. The term *top-n* is used to refer to a prediction set of size n, $|\mathbb{P}| = n$ where $n = \{1, 2, 3\}$. In the case of top-1, a correct prediction means that the attacker performed an action that corresponded to the symbol identified by the model as the most likely to occur. Similarly, a correct prediction in terms of *top-2* and *top-3* mean that the attacker's action is among the two and three most probable attack steps identified by the model.

B. Experiment 1: Effectiveness of VLMM

Figure 6 shows the prediction results of different Markov Models created from sequences mapped based on attack descriptions (Ω_d). The dash ('-') represent the prediction rate when considering the symbol that is attributed the highest probability of occurrence (top-1). Points marked with '$\times$' and '*' represent top-2 and top-3 prediction rates, respectively. Results from ten independent runs with random 50-50 split training and testing sets are averaged and shown with one standard deviation.

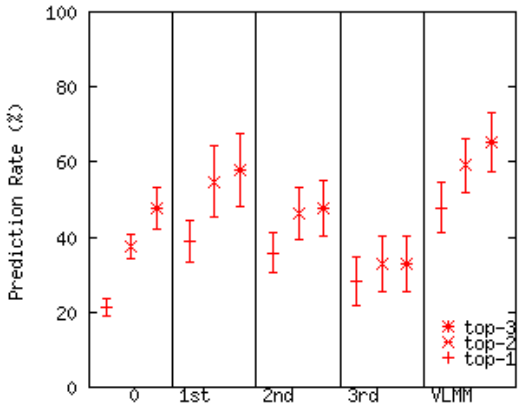

Fig. 6. Prediction rate using 0^{th}, 1^{st}, 2^{nd}, and 3^{rd} order models on Ω_d.

As mentioned in Section III, the relationship between an attack action and the attacker's recent behavior can be inferred from the correlation of neighboring actions within a sequence. When using a first order model, the next attack action is predicted based on the immediate previous action. Similarly, an n^{th} order finite-context relates the next action with the n previous ones. For the results in Figure 6, attack actions were predicted based on 0, 1^{st}, 2^{nd}, 3^{rd} order finite-context models as well as a VLMM. For example, when top-3 is considered, the correct prediction is about 48% of the time in the case of a 0-order predictor, 59% for a 1^{st}, 48% for 2^{nd}, 35% for 3^{rd}, and 68% for the VLMM.

The 0-order model, which predicts based on frequency counts of previously observed attacks, yields performance

[2]A second data-set was created for a different network topology and tested. It exhibits similar performance trends as the ones detailed in this paper.

among the lowest. This suggests the importance of an attacker's previously observed actions when inferring his or her next step. The 1^{st} order model performs better than the 2^{nd} and the 3^{rd} order models, indicating that the next exploit has a strong correlation with the attacker's immediate previous action. This is intuitively correct, as the immediate previous action often grant certain privileges for the attacker's next action. Also notice that VLMM outperforms the 1^{st} order model, which suggests that, while higher order models individually may not be a good indicator, they do introduce additional information that is not captured by the 1^{st} order model. The end result for the combined VLMM prediction is a top-3 accuracy approaching 70% when predicting specific methods the attacker may use next during an ongoing attack.

Sequences can also be created based on attack categories (Ω_c). Figure 7 plots the prediction accuracy for the case of Ω_c. The results are impressive; the VLMM achieves 90% accuracy when predicting the next attack *type* in an ongoing attack sequence. The finer the granularity of a data model, the more information is needed in order for special cases to be captured with statistical significance; therefore, with the same data-set, finer grained models are expected to have worse prediction rates. This can be noticed by comparing Figures 6 (Ω_d) and 7 (Ω_c). Arguably, predicting at coarser levels of granularity may also be advantageous given the dynamic nature of exploitations. Since specific exploitation methods can evolve due to configuration changes and discovery of new vulnerabilities, modeling and predicting at coarser levels of granularity may be more desirable because it provides network analysts with overall directions instead of specific yet not necessarily accurate predictions.

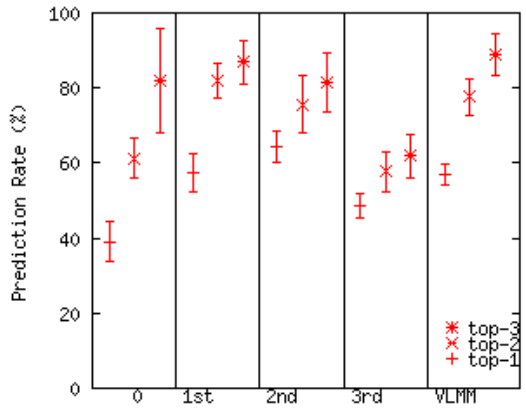

Fig. 7. Prediction rate using 0^{th}, 1^{st}, 2^{nd}, and 3^{rd} order models on Ω_c.

Since attack behavior can be very diverse, certain attack sequences may be easier to predict than others. Information entropy is a logical choice to measure the confidence or uncertainty of each predicted action. Let H_i be the entropy of the i^{th} symbol of $s = \{x_1, x_2, ..., x_m\}$ given a prediction model. The larger the value for H_i, the larger the uncertainty associated with the attack step X_i.

$$H_i = -\sum_{x_i \in \Omega} P\{X_i = x_i \mid X_1 = x_1, ..., X_{i-1} = x_{i-1}\} \times \log_2 P\{X_i = x_i \mid X_1 = x_1, ..., X_{i-1} = x_{i-1}\} \quad (4)$$

H_i was calculated for all symbols in the testing data given a VLMM. The values of H_i were combined into two categories depending on whether the symbol was correctly or incorrectly predicted. Results are presented in Table I. Notice that, despite the large standard deviation, symbols that were mispredicted have, on average, higher entropy. Further investigation will help us understand this large fluctuation.

TABLE I
ENTROPY OF CORRECTLY PREDICTED AND MISPREDICTED ATTACK STEPS FOR DIFFERENT ATTACK ACTION SUBSPACES.

	Category (Ω_c)	Category IP (Ω_t)	Description (Ω_d)
c	0.62 ± 0.48	$.91 \pm .60$	1.07 ± 0.69
i	0.93 ± 0.63	$1.41 \pm .81$	1.35 ± 0.71
c	correct predictions		
i	incorrect or mispredictions		

In addition to entropy, average log-loss was used to measure the complexity of an entire attack sequence. The greater the probability assigned by a predictor to the symbols of a sequence, the smaller is the sequence's average log-loss. In this work, average log-loss is used to classify a sequence as *common* or *sophisticated*. Let P be a predictor that assigns a probability $P(x_{i,j})$ to symbol $x_{i,j}$. Then, given a sequence $s_i = \{x_{i,1}, x_{i,2}, ...x_{i,m}\}$, the average log-loss is defined as

$$l(P, s_i) = -\frac{1}{m}\sum_{j=1}^{m} \log P(x_{i,j} | x_{i,1}, ..., x_{i,j-1}) \quad (5)$$

For the results presented on Table II, the VLMM was used as the predictor P, and the average log-loss of each sequence in the test-set was computed based on (5). A threshold value of T is used when classifying attack sequences as common $l(P, s) \leq T$, versus sophisticated $l(P, s) > T$. Table II shows the prediction rates of common and sophisticated sequences classified based on different values of T chosen empirically. Notice that results based on different attack action subspaces are given. When comparing the two sets, one would expect the higher log-loss sequences (sophisticated set) to have lower prediction rates. The results shown in Table II are in accordance with this intuition.

TABLE II
PREDICTION RATES OF COMMON AND SOPHISTICATED SEQUENCES CLASSIFIED BY THRESHOLDING THEIR AVERAGE LOG-LOSS.

	Threshold T	Category Ω_c	Category IP Ω_t	Description Ω_d
c	≤ 1.5	0.91	0.86	-
s	> 1.5	0.7	0.52	0.52
c	≤ 2.0	0.83	0.91	0.85
s	> 2.0	0.69	0.50	0.51
c	≤ 2.5	0.79	0.79	0.80
s	> 2.5	0.65	0.48	0.49
c	≤ 3.0	0.77	0.66	0.71
s	> 3.0	0.60	0.46	0.47
c	common sequences			
s	sophisticated sequences			

Sequences created based on attack description (Ω_d) and

attack category (Ω_c) model the transitions within attack methods chosen by an intruder. Another logical procedure is to characterize attack behavior with regards to how an attacker maneuvers within a network. This is possible when attack destination IP is incorporated into the model. Extending the use of Ω_c, a new set of sequences are created according to the tendency of an attack action to happen at the same or at a different target machine from the immediate previous action (Ω_t). The idea behind this approach is to examine the agility of each attack track.

When incorporating the destination IP into the model (Ω_t), we noticed that only 2% of the DoS attack actions were incurred on a different machine than the one targeted by the immediate previous action. DoS is an attempt to make a computer resource unavailable to its intended users by saturating the victim machine with external communications requests; therefore, this low tendency for DoS to cover several distinct machines within the same attack sequence is expected. On the other hand, 21.6% and 16.7% of the "data exfiltration" and "backdoor, trojan, virus and worm" attacks happened on a different target machine from their preceding attack actions. This suggests that "data exfiltration" and "backdoor, trojan, virus and worm" are more 'agile,' or more likely to target multiple machines in a network.

Figure 8 provides an insightful representation for these attack transitions. The x-axis represents the number of transitions in attack target within an attack sequence. The y-axis represents the number of unique target machines visited by the attack sequence; that is, the number of distinct destination IPs. For example, the point $(9, 3)$ represents a sequence in which the attacker hopped 9 times across 3 different machines. The size of the squares in Figure 8 are logarithmically proportional to the number of attack tracks that fall within the given point. For example, most of the attack sequences in our data lie in $(0, 1)$, which is represented by the large square. The $(0, 1)$ coordinate tells that the attacker targeted only one machine; therefore there are no transitions within his or her attack track.

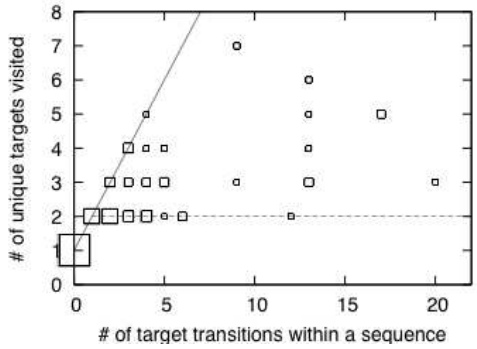

Fig. 8. The number of unique targets visited versus the number of target transitions within the attack sequences.

Figure 8 can be used to characterize attack behavior. Points close to the $y = 2$ line that have $x >> 1$ represent attackers that hop many times across a small number of machines. For example, point $(20, 3)$ represents an attack sequence that contained 20 transitions covering only three different destination IPs. On the other hand, points closed to or at the $y = x + 1$ line that have $x >> 1$ indicate attacks that hop across many distinct machines.

Given the differences in target IPs and IP transitions, one may hypothesize that the more 'agile' attack sequences are more 'sophisticated' and harder to predict. A threshold of $T = 2.5$ was chosen to classify roughly half of the sequences in the data-set as common and half as sophisticated based on their average log-loss. Then, the number of target IP transitions and the prediction rate for the sophisticated and the common attack sequences was computed as shown in Table III. Despite the large deviations, the significant difference in the average IP transitions (3.71 versus 0.50) suggests that the more agile multi-stage attacks are likely to exhibit rare ordering in attack actions and are harder to predict. Note that the approach presented in this paper allows the examination and classification of attack sequences based on target IPs without requiring information about network topology and system configurations.

TABLE III
AVERAGE NUMBER OF ATTACK TARGET TRANSITIONS FOR SEQUENCES THAT WERE CLASSIFIED AS COMMON AND SOPHISTICATED.

	Threshold (T)	# transitions per seq	Prediction rate
c	≤ 2.5	0.50 ± 1.40	0.79
s	> 2.5	3.74 ± 4.71	0.48
c	common sequences		
s	sophisticated sequences		

C. Experiment 2: Real-time adaptation

Performance results shown in Experiment 1 can be improved if the new attacks and attack sequences are incorporated into the model as they are observed. An implementation that takes correlated alerts, one by one, and continuously updates the model as it predicts is tested using the same VMware data-set. For this second experiment, the entire data-set is fed into the real-time system according to the time stamp of the alerts. This real-time implementation is possible because of the computational efficiency of the VLMM model and prediction.

Figure 9 shows the window average prediction accuracies (top-3) achieved by the system versus the total number of injected alerts for the Ω_d and Ω_c subspaces. The window size is approximately 100 alerts. This set of results reveals how the real-time system performs as the model is trained with more and more alert data. First, observe the initial transient periods where the system has not yet observed enough alerts to build an accurate model. After approximately 1,000 alerts, the prediction accuracies rise but fluctuate around 75% and 95% for Ω_d and Ω_c, respectively. The overall cumulative average prediction accuracies for the entire data-set are 76.25% and 94.69% for Ω_d and Ω_c, respectively.

As expected, the overall performance improves by continuously updating the model with new attacks and new attack sequences. The fluctuation observed can be partially attributed to the introduction of new symbols in the corresponding alphabet. Figure 10 shows the percentage of symbols known to the model as the alerts are injected. Note that with 1,000 alert injected, only 36% and 67% of symbols have occurred in

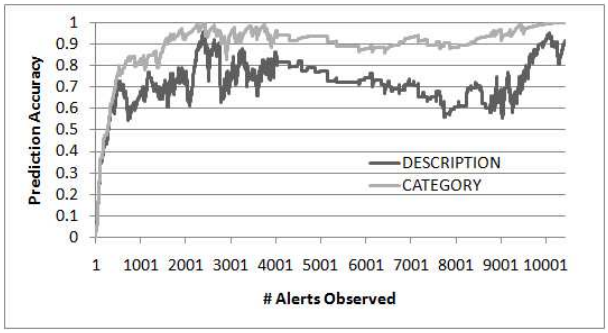

Fig. 9. Window average prediction accuracy (top-3) as alerts are injected to the real-time system for the cases of Ω_d and Ω_c.

Ω_d and Ω_c, respectively. In fact, because the alerts are injected according to their time stamps, which correspond to the attack scenario they belong to, new symbols may not be seen by the system until late in the simulated time period. Even with new symbols being continuously introduced by different attack scenarios, the system prediction accuracies remain between $55\% \sim 95\%$ and between $85\% \sim 100\%$ for Ω_d and Ω_c, respectively.

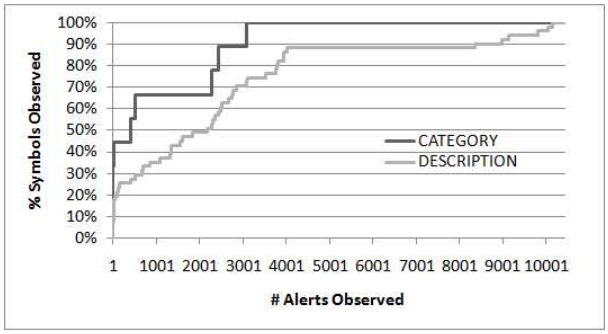

Fig. 10. Percentage of Symbols Observed as alerts are injected to the real-time system for the cases of Ω_d and Ω_c.

The number of symbols generally levels during the middle of the five scenario sets, but each scenario contributes with new symbols to the *description* alphabets. The last scenario is an interesting case. It starts after ∼8,000 alerts have been observed and it introduces new specific attack methods to the model as noticeable from Figure 10. For this reason, the system performance drops temporarily. After 1,000 additional alerts have been observed, the system performance rises again to around 85%-90% accuracy. This phenomenon suggests the adaptiveness of the proposed approach in the presence of new attacks. Note that the new attacks can be introduced by a combination of changes in network configuration and the discovery of new exploits.

Note also that while the prediction performance fluctuates for Ω_d in the beginning of the last scenario, the use of Ω_c maintains a steady performance of 95%. This is encouraging and reinforces the notion that assessing at a coarser level of granularity helps the analysts focus on the right type of attacks (as opposed to relying on software to suggest specific attacks that may not have been observed with enough frequency in order to yield accurate predictions). In fact, given his or her a priori knowledge and experience on the specific network, an analyst should probably know more about new (zero-day) attacks than the system does.

The real-time implementation also attempts to predict that a never-before-seen attack is about to happen. The system does not predict exactly what new attack will happen; instead, it predicts that *something new* will occur. In addition to the regular symbols representing specific attack methods or attack categories, each alphabet subspace contains a special symbol to track the instances when a new attack happens for the first time. When these instances occur, the suffix tree will update for the branches leading to this symbol. Although there are few such instances, incorporating this symbol in the VLMM allows warning to be generated for new attacks. In our experiment, the system is able to predict 18 out of 51 occurrences of new attack methods ($\Omega_d = 51$). While $18/51 \approx 35\%$ is not as impressive as the prediction accuracy presented earlier, this result shows promises that VLMM is able to not only adaptively train and predict attacks that have been observed, but also provide warnings of new attacks.

V. Conclusion

This paper introduces a framework for the characterization and prediction of cyber attack behavior. Built upon existing technologies, namely, IDSs and alert correlation engines, the proposed approach aims at capturing the sequential properties residing in the correlated IDS alerts. Different from existing approaches that heavily rely on network specific information, our approach does not require the modeling of network configuration and system vulnerabilities. This separation is accomplished by applying ideas from universal predictors on high-level attack information. Specifically, the behavior trends exhibited in various fields of IDS alerts are captured via VLMMs. Our results demonstrate that sequential properties, *i.e.*, , the 1^{st}, 2^{nd}, 3^{rd},... order Markov models, are all beneficial, and a combination of them via VLMM leads to the best prediction accuracy. Information theory based metrics, such as entropy and log-loss, are proposed as indicators of the prediction quality.

Historical and ongoing alert sequences are used to build and update suffix tree models that store statistical information for VLMM predictions. Since it does not require specific network information, the proposed methodology is able to adapt to new attacks and new attack sequences, regardless of changes in network configurations or discoveries of new exploits. Our results demonstrate that, soon after new attack scenarios are introduced, the prediction accuracy rises as the model starts capturing sufficient statistics for the new attacks and attack sequences. The new attack predictions will not be diluted by the historical sequences as the new symbols and new symbol sequences are different from the old ones.

We believe that the battle against cyber attacks goes beyond password protection, encryption, intrusion detection, and alert correlation. Having these components is essential for protected network operations and usage; however, a proactive measure that projects ongoing attack actions is crucial for timely mitigation of cyber attack impacts. In order to create a comprehensive assessment of cyber attacks, we propose the

presented approach to be considered as part of the cyber attack projection solution, complementing the projection schemes that depends on network specific information.

ACKNOWLEDGMENT

The authors would like to thank Jared Holsopple, Moises Sudit, John Salerno, George Tadda and Michael Hinman for their valuable inputs. This work is funded through the National Center for Multisource Information Fusion (NCMIF) grant under the technical supervision of AFRL/IFEA.

REFERENCES

[1] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers University, 2000.

[2] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *The International Journal of Computer and Telecommunications Networking*, 31:805–822, 1999.

[3] Tim Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43:99–105, 2000.

[4] John R. Goodall, Wayne G. Lutters, and Anita Komlodi. The work of intrusion detection: rethinking the role of security analysts. In *Proceedings of the Americas Conference on Information Systems*, pages 1421–1427, 2004.

[5] Frédéric Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the* 17^{th} *Anual Computer Security Applications Conference*, pages 22–31, 2001.

[6] Oliver Dain and Robert K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of ACM Workshop on data mining for security applications*, November 2001.

[7] Samuel King, Morley Mao, Dominic Lucchetti, and Peter Chen. Enriching intrusion alerts through multi-host causality. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2005.

[8] Peng Ning, Yun Cui, and Douglas Reeves. Analyzing intensive intrusion alerts via correlation. In *Proceedings of the* 9^{th} *ACM Conference on Computer & Communications Security*, 2002.

[9] Adam Stotz and Moises Sudit. INformation Fusion Engine for Real-time Decision making (INFERD): A perceptual system for cyber attack tracking. In *Proceedings of the International Conference on Information Fusion*, 2007.

[10] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 2212, pages 54–68, 2001.

[11] Frederik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 01(3):146–169, 2004.

[12] André Arnes, Frederik Valeur, and Richard Kemmerer. Using hidden markov models to evaluate the risk of intrusions. In *Proceedings of the International Symposium of the Recent Advances in Intrusion Detection (RAID)*, Hamburg, Germany, 2006.

[13] Jarred Holsopple, Shanchieh J. Yang, and Moises Sudit. TANDI: Threat Assessment of Network Data and Information. In *Proceedings of SPIE Security and Defense Symposium: Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, 2006.

[14] Zhitang Li, Jie Lei, Li Wang, and Dong Li. Assessing attack threat by the probability of following attacks. In *Proceedings of the International Conference on Networking, Architecture, & Storage*, pages 91–100, 2007.

[15] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. In *Proceedings of the* 20^{th} *Annual Computer Security Applications Conference*, pages 370–379, 2004.

[16] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund M. Clarke, and Jeannette Wing. Ranking attack graphs. In *Proceedings of the International Symposium on the Recent Advances in Intrusion Detection (RAID)*, September 20-22 2006.

[17] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical approaches to DDoS attack detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, volume 1, pages 303–314, 2003.

[18] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service activity. *ACM Transactions on Computer Systems*, pages 115–139, 2006.

[19] Kihong Park and Heejo Lee. On the effectiveness of probabilistic packet marking for IP traceback under Denial of Service Attack. In *Proceedings of IEEE INFOCOM*, pages 338–347, 2001.

[20] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for Unix processes. In *Proceedinges of IEEE Symposium on Research in Security and Privacy*, pages 120–128, 1996.

[21] Calvin Ko, Manfred Ruschitzka, and Karl Levitt. Execution monitoring of security-critical programs in distributed systems: a specification-based approach. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 175–187, 1997.

[22] Andrew Kosoresow and Steven Hofmeyr. Intrusion detection via system call traces. *IEEE Software*, 14:35–42, Sep/Oct 1997.

[23] Wenke Lee, Salvatore J. Stolfo, and Philip K. Chan. Learning patterns from Unix process execution traces for intrusion detection. In *Proceedings of Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56, 1997.

[24] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusion using system calls: Alternative data models. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 133–145, 1999.

[25] Nong Ye, Xiangyang Li, Qiang Chen, Syed M. Emran, and Mingming Xu. Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man and Cybernetics*, pages 266–274, 2001.

[26] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the* 9^{th} *ACM Conference on Computer and Communications Security*, pages 255–264, 2002.

[27] Wen-Hua Ju and Yehuda Vardi. A hybrid high-order markov chain model for computer intrusion detection. Technical Report 2, National Institute of Statistical Sciences, 1999.

[28] Terran Lane and Carla Brodley. Temporal sequence learning and data reduction for anomaly detection. In *ACM Transactions on Information and System Security*, volume 2, pages 295–331, 1999.

[29] Nong Ye, Yebin Zhang, and Connie M. Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53:116–123, 2004.

[30] Andrzej Ehrenfeucht and Jan Mycielski. A pseudorandom sequence – how random is it? *American Mathematical Monthly*, 99:373–375, 1992.

[31] Philippe Jacquet, Wojciech Szpankowski, and Izydor Apostol. A universal predictor based on pattern matching. *IEEE Transactions on Information Theory*, 48:1462–1472, 2002.

[32] M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38:1258–1270, July 1992.

[33] Daniel S. Fava. Characterization of cyber attacks through variable length markov models. Master's thesis, Rochester Insitute of Technology, 2007.

[34] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice Hall, 1990.

[35] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[36] Cosma R. Shalizi and Kristina L. Shalizi. Blind construction of optimal nonlinear recursive predictors for discrete sequences. In *Proceedings of the* 20^{th} *Conference on Uncertainty in Artificial Intelligence*, pages 504–511, 2004.

[37] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[38] Massachusetts Institute of Technology Lincoln Laboratory. DARPA intrusion detection evaluation, 2001. `http://www.ll.mit.edu/IST/ideval/data/data_index.html`.

[39] KDD Cup. KDD Cup data, 1999. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[40] DEFCON conference. DEFCON capture the flag (CTF) contest. `http://www.defcon.org/`.