Architectures for Detecting Interleaved Multi-stage Network Attacks Using Hidden Markov Models

Tawfeeq Shawly, Member, IEEE, Ali Elghariani, Member, IEEE, Jason Kobes, Member, IEEE, and Arif Ghafoor, Fellow, IEEE

Abstract—With the growing amount of cyber threats, the need for development of high-assurance cyber systems is becoming increasingly important. The objective of this paper is to address the challenges of modeling and detecting sophisticated network attacks, such as multiple interleaved attacks. We present the interleaving concept and investigate how interleaving multiple attacks can deceive intrusion detection systems. Using one of the important statistical machine learning (ML) techniques, Hidden Markov Models (HMM), we develop two architectures that take into account the stealth nature of the interleaving attacks, and that can detect and track the progress of these attacks. These architectures deploy a database of HMM templates of known attacks and exhibit varying performance and complexity. For performance evaluation, in the presence of multiple multi-stage attack scenarios, various metrics are proposed which include (1) attack risk probability, (2) detection error rate, and (3) the number of correctly detected stages. Extensive simulation experiments are used to demonstrate the efficacy of the proposed architectures.

Index Terms—cyber systems, network security, intrusion detection, Hidden Markov Model, interleaved attacks.

1 INTRODUCTION

Large organizations face a daunting challenge in the provision of security for their cyber-based systems. Modern cyber-based infrastructures typically consist of a large number of interdependent systems and exhibit increasing reliance on the security of such systems. In the present threat landscape, network attacks have become more advanced, sophisticated and diversified, and the rapid pace of coordinated cyber security crimes has witnessed a massive growth over the past several years. For instance, in May 2017, the “WannaCry” ransomware attack was detected after it locked up over 200,000 servers in more than 150 countries [1]. A month later, another version of the same attack caused outages of most of the government websites and several companies in Ukraine, and eventually, this attack spread worldwide [2]. With the explosive growth of cyber threats, a dire need exists for the development of high-assurance and resilient cyber-based systems. One of the most important requirements for high-assurance systems is the need for advanced and sophisticated attack detection and prediction systems [3].

Security reports reveal that, over time, the type of network intrusions have transformed from the original Trojan horses and viruses into more complex attacks comprised of a myriad of individual attacks. These attacks follow a series of long-term steps and actions referred to as multi-stage attacks, and therefore are hard to predict [4], [5]. During these attacks, an intruder launches several actions, which may not be performed simultaneously, but are correlated in the sense that each action is part of the execution of previous ones and each multi-stage attack is aimed at a specific target. The detection of multi-stage attacks poses a daunting challenge to the existing threat detection techniques [3]. This challenge is exacerbated when multiple attacks such as these are launched simultaneously in the network, originated by a single or multiple attackers trying to stealth certain attacks among others [5], [6].

1.1 Related Work

In the past, various approaches have been proposed to address intrusion detection challenges related to multi-stage attacks. These approaches can, in general, be categorized as correlation-based techniques [7]–[9] or machine learning (ML) based techniques. Examples of ML techniques include Hidden Markov Models, Bayesian Networks, Clustering and Neural Networks [10]–[13].

Correlation-based techniques, based on cause and effect relationships, mainly utilize attack-graphs when searching the possible stages of the attack [14]–[19]. For example, the work in [15] focuses on the causal relationships between attack phases on the basis of security information. Onwubiko et al. [16] assesses network security through mining and restoring the attack paths within an attack graph. A causal relations graph presented in [17], contains the low-level attack patterns in the form of their prerequisites and consequences. In this approach, during the correlation phase, a new search is performed upon the arrival of a new alert. Several other techniques use similar ideas for analyzing attack scenarios from security alerts [18], [19]. However, most of these approaches depend on correlation rules in conjunction with the domain knowledge. Due to increased computational complexity...
in detecting real time attacks, these techniques pose a limitation.

In the category of ML techniques, HMM is a leading approach for the prediction of multi-stages attacks [20–28]. In this approach, stages of an attack are modeled as states of the HMM. The HMM is considered the most suitable detection techniques for such attacks for several reasons [20]. First, it has a tractable mathematical formalism in terms of the analysis of input-output relationships, and the generation of transition probability matrices based on a training dataset. Second, because of its specialized capacity to deal with sequential data by exploiting transition probability between states, it can track the progress of a multi-stage attack. Despite the existing research in the use of HMM for intrusion detection in general and multi-stage attacks in particular, none of these approaches considers the problem of interleaving multi-stage attacks and analyzing the impact on the detection performance of such attacks. Moreover, existing approaches address only a single multi-stage attack.

1.2 Contribution

This research addresses several challenges to the detection of interleaved attacks which include: (1) how to model each multi-stage attack in terms of HMM states in the presence of mixed observations, (2) how to detect a multi-stage attack when an attacker(s) performs interleaved attacks with the intention to hide an attack (i.e. stealthy attacks), (3) since no standard public dataset is available that can provide interleaved traffic from simultaneous multiple attacks, the generation of this type of datasets poses a challenge to the research community, (4) the design of an efficient architecture that can detect and track the progress of multiple simultaneous attacks, and (5) the development of an approach to accurately quantify and measure the detection performance of such an architecture.

To address the above challenges, we propose in this paper two architectures based on HMM formalism. The proposed architectures exhibit varying detection performance and processing complexities. The architectures can detect the occurrence of multiple organized attacks and provide insights into the dynamics of these attacks such as identifying which attack is progressing and which one is idle at any point of time, how fast or slow each attack is progressing, and in which security state each attack is occurring at any given point in time. Knowledge of this information can assist in designing effective response mechanisms that can mitigate security risks to the network [3, 29]. The design of the first proposed architecture relies on modifying HMM model parameters to detect multiple multi-stage attacks in the presence of mixed alerts. The design of the second proposed architecture relies on de-interleaving mixed alerts from different attacks prior to the HMM processing subsystem. We compare the two architectures in terms of their detection performance and design complexity.

The remainder of this paper is organized as follows. In Section 2, we discuss how HMM is used to detect multi-stage attacks. We introduce the system model in Section 3. We present the proposed architectures in Section 4 and evaluation and performance measures in Section 5. We conclude the paper in Section 6.

2 THE HIDDEN MARKOV MODEL (HMM) FOR DETECTING MULTI-STAGE ATTACKS

An HMM is a double-stochastic process [30] in that it has an underlying stochastic process that is hidden, and can only be observed through another set of stochastic processes that produce the sequence of observed symbols. The observation process corresponds to alerts generated by the IDS. Mathematical preliminaries for the discrete HMM in the context of multiple multi-stage attacks are given in Appendix A.

2.1 Using HMM to Detect Multi-stage Attacks

In a multi-stage attack, an intruder launches a series of long-term steps and actions that are sequentially correlated in the sense that each action follows the successful execution of the previous one. In other words, the output of one stage serves as the input to a subsequent stage. One example of multi-stage attacks is the DDoS attack in which the attacker starts by scanning the targeted network in order to identify potential vulnerabilities. Subsequently, the attacker tries to break into vulnerable hosts which have been compromised by the attacker. After exploiting these hosts, the attacker installs a software such as a Trojan horse. Eventually, the attacker initiates access to the final target, which could be a server accessible from all the exploited hosts, and subsequently, the DDoS attack is launched [31].

Most detection systems have the capability to detect a single-stage attack or each of the stages of a multi-stage attack independently. However, the detection of multi-stage attacks poses a daunting challenge to the existing intrusion detection techniques due to the lack of an ability to analyze the entire attack activity chain as a whole. This challenge is exacerbated if several of these attacks are launched simultaneously in the network, each attack originated by a single or multiple attackers trying to stealth certain attacks within others. The difficulty in detecting interleaved attacks comes from the unrelated observations made of unrelated attacks, observations that conceal the details of the activity chains of multi-stage attacks.

Fig. 1 illustrates this challenge by exhibiting three possible scenarios involving the interleaving of three multi-stage attacks that can target a specific or multiple servers. For example, Attack 1, shown in yellow, is an SQL injection attack, wherein Attack 2, in orange, is a Brute force SSH, and Attack 3, in red, is a DDoS attack [31]. The table in the lower-left corner of the figure shows the correspondence between the type of attack, Alert
ID, Alert type, and stages for some observations of the aforementioned attacks. In addition, on the right side of the figure, an HMM heatmap shows the estimation of the belief about the current state assuming there are five stages for each multi-stage attack. A darker color indicates a higher probability and a higher degree of certainty about the current state.

In this example, ICMP PING is a common observation between Attack 1 and Attack 2. Also, assume that the system is in State 4 of Attack 1 and the next alert(s) generated by the IDS is ICMP PING, which is an observation for State 1 in this attack. The ICMP PING observation could be originally generated from Attack 2, thus, in this case, the state estimation can be affected due to the uncertainty regarding the exact current state of the system caused by the unexpected ICMP PING observation(s).

Fig. 1 also exhibits an example of how the degree of interleaving among the observations of three multi-stage attacks can hypothetically affect the performance of the state estimation over time. For instance, Scenario C in Fig. 1 has a higher degree of interleaving compared to Scenario B and, consequently, the uncertainty about the current state for each multi-stage attack at time t in Scenario C can be higher than in Scenario B. A detailed performance analysis regarding the interleaving is given in Section 5.

In this paper, we use HMM to model and detect possible multi-stage attack scenarios on a targeted cyber-based system. In particular, in order to detect a single multi-stage attack, (say Attack k), stages of the attack are modeled as states of the HMM and the observation process corresponds to related alerts generated by the IDS and processed later by a preprocessing component. Note, the aforementioned three multi-stage attacks can consist of different types in terms of the order of sequences and the number of stages and corresponding observations. Each attack type (k) is modeled using a distinct HMM template $\lambda_k$. In the case of $M$ possible attacks, we have a set of M templates.

Note, the selection of the optimum number of states for each HMM template is a challenge, and no simple theoretical answer exists as to how, in general, this parameter can be selected; this selection depends on the application [30]. In this paper, we model the number of HMM states so that they are similar to the number of stages of the multi-stage attack. The justification for this approach is that the closer the number of states is to the number of stages in the multi-stage attack, the better the details can be provided regarding the progress of the attack; therefore this approach can lead to the development of a more effective response mechanism.

Also note, for each attack type, multiple instances of the same type of attack can be launched by the attacker(s) and consequently, each instance constitutes a distinct attack. The distinction among instances is maintained by a set of observations features such as the source and destination IP addresses and ports. The full description of the attributes and features associated with observations is given in Section 4.

The parameters of the HMM model (i.e. the HMM model $\lambda_k$) for the multi-stage attack $k$ include the number of states of its Markov chain, the number of related IDS observations and aforementioned probability matrices A and B. These parameters are derived offline from a training dataset that contains alerts of a similar multi-stage attack scenario and which can be reestimated and improved online [32]. Specifically, each state is trained based on the observations that belong to the corresponding stage. Subsequently, in the presence of observations related to Attack $k$, HMM estimates the probability of being in each state of the model using Viterbi Algorithm [30]. However, as mentioned earlier, in the presence of multiple interleaved multi-stage attacks, the performance of the state estimation degrades significantly, especially in a scenario that contains a high degree of interleaving among the observations of multi-stage attacks. In the next section, we discuss interleaved multi-stage attacks in detail and present an HMM-based architecture.

3 System Model and Architecture

In order to detect multiple multi-stage attacks, say $K$ attacks, one can generalize the existing single attack architecture by building a database of $K$ HMM templates. In Fig. 2, we present a generic architecture for the threat detection process that uses such a database. Here, each HMM-based template is designed to detect a specific type of multi-stage attack. The goal of this generic architecture is to detect $K$ multi-stage attacks originated from a single or multiple attackers. Note, each of the $K$ HMM templates is trained to detect an individual multi-stage attack. As mentioned earlier, each template encompasses the HMM structure including all its parameters.

The second major component of this architecture is the Intrusion Detection System (IDS), (e.g., Snort software...
which generates the attack related alerts in real time from the network traffic according to a predefined set of rules. Typically, an IDS generates a stream of alerts which are temporally ordered based on their timestamps. The online processing of this stream of alerts can potentially require a large amount of memory. Such memory requirements can be improved by implementing Snort rules using deterministic or nondeterministic forms of finite automata. The selection of IDS rules can help to reduce the large volume of alerts and false positives by tuning these rules. The interleaved alerts generated by Snort can belong to one or multiple attacks. These alerts can be preprocessed to generate observations in a suitable format that can be forwarded to the HMM database. Based on the information from Snort, the preprocessing module can assign different severity levels for the incoming alerts. The higher the level is, the more severe the alert which indicates an ongoing multi-stage attack is progressing towards an advanced stage. In this paper, we assume a window-based technique which is needed in order to buffer a finite number of observations so that these can be processed by the HMM templates. For this purpose, the incoming alerts to the system are grouped together to form an observation sequence of window size (observation length) \( T \). We assume no overlap occurs between two consecutive windows. Note, the risk of progressing multi-stage attacks can be assessed in real time by the risk assessment component. Prioritized response actions can be taken based on detected states and the risk of the active attacks.

### 3.1 Modeling Interleaved Attacks

Note, in general, \( K \) distinct multi-stage attacks can be launched simultaneously in a network, and their related alerts, generated by the IDS (Snort), are forwarded to the HMM database in the form of a single stream of interleaved alerts. These alerts can be the result of a systematic interleaving of multiple multi-stage attacks initiated by a single attacker or can be generated randomly by different attackers. Note, for each observation length \( (T) \), we assume \( T \) alerts are processed by the HMM templates sub-system. In particular, at any time, it is possible that these \( T \) alerts can result from one attack or a mix of at most \( K \) attacks. Some possible interleaved attack scenarios that can be orchestrated by an attacker include:

- An attacker starts and finishes an attack (Attack 2) in the middle of another ongoing attack (Attack 1) as shown in Scenario A in Fig. 1.
- Multiple attacks start and finish at different times in the presence of one or multiple ongoing attacks.
- Stages of attack(s) can be embedded at different times of an ongoing attack(s).
- Systematic interleaving among multiple multi-stage attacks can be launched based on interleaving groups of alerts (see; for example, Scenario C in Fig. 1).

The existing datasets which feature multi-stage attacks and are publicly available, do not consider these complex attack scenarios. The DARPA2000 alerts dataset, for instance, contains two distributed denial-of-service (DDoS) multi-stage attacks that happened at different times in which the attacker used multiple distributed compromised hosts to launch DoS attacks on a specific target. To address the challenge of generating the aforementioned interleaved attack scenarios, we generate interleaved alerts by altering timestamps and IP addresses of the DARPA2000 dataset.

In order to detect the aforementioned attack scenarios, we propose two architectures based on the generic architecture shown in Fig. 2. The design of the first architecture, Architecture I, is based on modifying the HMM model parameters so that they can deal with the interleaved alerts. The design of Architecture II improves attack detection capability by separating alerts from the various attacks prior to routing the alerts to HMM templates sub-system.

### 4 Proposed Architectures

#### 4.1 Proposed Architecture I

As mentioned earlier, in order to deal with interleaved traffic alerts from different attacks, we modify the HMM of the generic architecture to accommodate observations from different attacks. The modified architecture is shown in Fig. 3. The stream of alerts generated by the IDS contains alerts that belong to one or more concurrent attacks. That is, for each observation length \( T \), there are \( T \) observations \( (o_1, o_2, \ldots, o_t, \ldots, o_T) \) processed by the HMM detection system, as shown in Fig. 3. Arrival of these alerts represents the interleaved attacks mentioned in Section 3.2. The HMM template is trained for Attack \( k \). Therefore, out of \( T \) observations, HMM \( k \) is expected to distinguish and process only those observations that belong to its attack, for which this HMM has been designed. Note, among \( T \) observations, there are \( L_k \)
observations (i.e., \( \{ o_{1k}, o_{2k}, \ldots, o_{L_k} \} \)) belonging to Attack \( k \), and the HMM \( k \) consider the remaining \( T - L_k \) observations to be unrelated (interfering) alerts. We introduce a common state that encompasses all of the unrelated alerts in HMM \( k \).

Regarding HMM structure, we focus on the alerts generated by the IDS which may consist of True Positive (TP) and False Positive (FP) observations. For each template, we consider State 1 as the most likely state that can be inferred by observing \( T - L_k \) unrelated observations using HMM \( k \). In other words, the occurrence of these interfering (unrelated) observations leads to the lowest security state (State 1) in the HMM \( k \). To deal with these unrelated observations in parameterizing HMM \( k \), we introduce a new symbol, \( \{ o_t \notin V_k \} \), that represents all unrelated observations for Attack \( k \). This requires modifying the HMM parameters, (i.e., matrices \( A_k \) and \( B_k \)). This modification can be obtained by considering an observation \( o_t \), such that \( o_t \notin V_k \). Therefore, we add an extra column in the emission probability matrix, \( B_k \), to account for this new symbol, as follows:

\[
B_k = \begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1M_k} & \epsilon_1 \\
b_{21} & b_{22} & \cdots & b_{2M_k} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
b_{N_k1} & b_{N_k2} & \cdots & b_{N_kM_k} & 0 
\end{bmatrix}
\]

Note, transition to State 1, in the presence of unrelated observation \( o_t \), occurs with probability \( \epsilon_1 \) which has a very small value (such as \(< 1 \times 10^{-6}\)) chosen such that \( \sum_{j=1}^M b_{1j} = 1 \). Accordingly, almost no change is made to the other observation probabilities in the first row of the emission probability matrix. In addition, setting the probability to zero in the rest of the last column increases the probability that observing \( \{ o_t \notin V_k \} \) leads to State 1. A second modification is needed for the transition probability matrix \( (A_k) \) to ensure that whenever HMM \( k \) observes the \( T - L_k \) alerts from attacks other than Attack \( k \), transition to State 1 occurs. This transition can be achieved by introducing transition probability \( (\epsilon_2) \) in the first column of the \( A_k \) matrix. Although our initial assumption is based on a left-right model, in this architecture, instead of adding a new state to the model we let all other states return only to State 1 whenever alerts from unrelated attacks occur. An important advantage of modeling unrelated alerts in this way is that the training of each HMM is simplified. Subsequently, by introducing \( \epsilon_2 \), the matrix \( A_k \) becomes:

\[
A_k = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1N_k} \\
\epsilon_2 & a_{22} & \cdots & a_{2N_k} \\
\vdots & \vdots & \ddots & \vdots \\
\epsilon_2 & 0 & \cdots & a_{N_kN_k} 
\end{bmatrix}
\]

Based on this modification and training of the HMM template \( (\lambda_k) \), the evaluation module determines whether Attack \( k \) is active or not, as shown in Fig. 3 according to the criteria \( Pr(O|\lambda_k) \geq thr \). Note, \( thr \) is a threshold used to avoid unnecessary computations of the Viterbi algorithm module in case the attack is not active. The \( thr \) value can be chosen within a range of 0 to 0.5. However, with the larger the value of \( thr \), the HMM template \( (\lambda_k) \) estimates only the states of the high probability sequences. In this paper, we take a conservative approach in choosing \( thr = 0 \). The evaluation probability can be computed using the forward algorithm [30]. In case the Attack \( k \) is active, then HMM \( k \) runs the Viterbi algorithm to decode the most probable hidden states that correspond to the given observation sequence \( O = \{ o_1, o_2, \ldots, o_t \} \), as follows:

\[
x_t = \max_{1 \leq i \leq N_k} \gamma_t(i) \]

\[
\gamma_t(i) = Pr(x_t = s_i|O, \lambda_k) \quad (1)
\]

where \( \gamma_t(i) \) represents the probability of being in state \( s_i \) at time \( t \) based on the observation sequence. In Architecture I, each HMM template in Fig. 3 uses the Viterbi algorithm to find the best state sequence, \( X = \{ x_1, x_2, \ldots, x_T \} \). For a given observation sequence, the Viterbi algorithm finds the highest probability along a single path for every \( o_t \) (\( t \leq T \)) such that:

\[
\delta_t(i) = \arg\max_{s_1, \ldots, s_{t-1}} Pr(s_1, \ldots, s_t, o_1, \ldots, o_t|\lambda_k) \quad (2)
\]

Using induction, the algorithm determines the rest of the state sequence, as follows:

\[
\delta_{t+1}(j) = \arg\max_{1 \leq i \leq N_k} \delta_t(i)a_{ij}(k).b_i(o_{t+1}(k)) \quad (3)
\]

This computation for a given sequence is repeated by all HMM templates in the Architecture I (Fig. 3). Table 1 shows the overall processing of alerts based on Architecture I.

Architecture I has the limitation of a high probability of high false negatives in states detection, especially when the attacks are highly interleaved as shown in Section 5. The reason for this limitation in such scenarios is that each HMM template processes an observation sequence that contains interfering observations belonging to other attacks. However, the low performance of Architecture I is observed only in special attack scenarios. Nevertheless, Architecture I has a low computation
complexity in terms of observations preprocessing (as mentioned earlier). More details about the detection performance of Architecture I is given in Section 5.

To achieve better performance, we propose another variation of the generic architecture of Fig. 2. Tabled as Architecture II, this new architecture, is depicted in Fig. 4 and is discussed below.

TABLE 1: Detection process for Architecture I

| Input: interleaved alerts: \( O = \{o_1, o_2, \ldots, o_T\} \), \( \pi_L\), \( A_k \), \( k = 1, 2, \ldots, K \), \( \lambda_k \) \|
|---|---|
| 1. Preprocessing: \( X = \{x_1, x_2, \ldots, x_T\} \) \|
| 2. While \( (O \) is not empty) \|
| 3. \begin{align*}
| & \text{for } k = 1 : K \\
| & \text{if } (Pr(O|\lambda_k) \geq (tr)) \\
| & \text{for } i = 1 : T \\
| & \text{Compute } \gamma_i(i), \text{ from equation (1)} \\
| & x_i = \max_{1 \leq i \leq N_k} \gamma_i(i) \\
| & \text{endfor} \\
| & \text{endif} \\
| & \text{endfor} \\
| & \text{endWhile} \\
|---|

4.2 Proposed Architecture II

Again, we consider \( K \) interleaved multi-stage attacks that can be simultaneously launched in the network. The IDS system, based on Snort, generates alerts from these attacks. Every alert is generated with a set of features, which includes alert ID, source/destination IP address, source/destination port number, and timestamp. In Architecture II, we use these features to improve detection efficiency of the HMM templates. In particular, unrelated

whether their instances are from the same or different types of attacks.

A subset, \( S \), from the feature set \( F \) \((S \subset F)\) can be used for the demultiplexing operation. The simplest way in which we can demultiplex interleaved alerts is by grouping the alerts that are triggered by the same multi-stage attack into one subsequence based on their IP addresses relationships, i.e., \( S = \{f_3, f_5\} \). Note, the IP addresses of the alerts, which are triggered by the same attack scenario, are generally related in forming a single substream. Consider there are two alerts, \( o_i \) and \( o_j \). The demultiplexer searches for their addresses to check if they have the same srcIP or the same desIP. Moreover, it also checks whether destIP of the previous alert is the same as the srcIP of the next one, as in a multi-stage attack scenario, as when the destination node of an earlier alert is the source node of the next alert. Based on the IP address search, the demultiplexer either inserts \( o_i \) and \( o_j \) in the same subsequence or in different ones.

In essence, the demultiplexer module demultiplexes the alert streams into \( L \) substreams \((1 \leq L \leq K)\). The demultiplexing process is based on one or more of the aforementioned distinguishing features of the incoming alerts and/or based on the correlation of IP addresses. Therefore, from the incoming stream of alerts, \( O = \{o_1, o_2, \ldots, o_T\} \), the demultiplexer generates \( L \) substreams each of which belongs to a distinct multi-stage attack. These substreams are a subset of \( O \), which are represented as \( O_1 = \{o_{1i}, o_{2i}, \ldots, o_{rt}\} \) \( O_k = \{o_{1k}, o_{2k}, \ldots, o_{rt}\} \), and so on till \( O_L = \{o_{1L}, o_{2L}, \ldots, o_{rt}\} \), where \( L \leq K \) and \( T_k \leq T \). Note, the larger the feature subset we consider in stream demultiplexing, the more distinct substreams we obtain and, in turn, the more processing is entailed. Note, within one observation sequence, alerts can belong to \( L \) attacks where \( 1 \leq L \leq K \). We assume that the demultiplexer does not cause any error in generating substreams.

The demultiplexer module does not distinguish among types of attacks, therefore, it cannot route a substream to its corresponding HMM template. To address this issue in Architecture II, each HMM can have \( L \) in-

...
In Architecture II has a maximum of $K$ operations. The larger the $K$ is, the more complex the computation is performed by this module. In other words, as a result of the demultiplexing operation, Architecture II has $T \times |S|$ additional computational steps as compared with Architecture I.

Next, we consider the HMM database component of the architectures. Note, two algorithms need to be executed in each branch of the HMM database, the forward algorithm (FW) to compute posterior probability for the evaluation purpose and the Viterbi algorithm (VA) to estimate the best state sequence. In Architecture I, each incoming sequence of $T$ alerts is processed by all of the $K$ branches. In other words, $K$ computations of the FW algorithm plus $K$ computations of the Viterbi algorithm are performed. On the other hand, in Architecture II, each HMM template processes, on the average, with a shorter sequence length compared to the sequence lengths in Architecture I. In the first module of each branch, the FW algorithm is executed $L$ times, and in the second module of the branch, the Viterbi algorithm is executed only once. Therefore, in Architecture II, $L \times K$ computations of the FW algorithm plus $L$ computations of the Viterbi algorithm are performed. It is important to note that although Architecture II seems to perform a greater number of computations in the HMM database, the length of sequences processed by both the FW and the Viterbi algorithms is, on the average, shorter than the sequences in Architecture I. The primary shortcoming of Architecture II is the computation overhead needed for the demultiplexing operation. This overhead can be high especially in cases where $T$ has a very large value and a large number of features.

### 5 Performance Evaluation

In this section, we discuss the experimental results based on the DARPA2000 dataset [31], since limited datasets are available for this particular evaluation. The DARPA2000 dataset contains two DDoS multi-stage attacks labeled as LLDDOS 1.0 and LLDDOS 2.0.2. Each of these attacks has five stages: 1) IP sweeping, 2) Sadmind probing, 3) Sadmind exploitation, 4) DDoS software installation, and 5) Launching the DDoS attack. In our experiment, DARPA2000 raw network packets were processed by Snort IDS [33] to generate alerts. The total number of alerts resulting from this process is 3500 and 2000, for LLDDOS 1.0 and LLDDOS 2.0.2 attacks, respectively. These alerts are clustered into 12 distinct symbols, therefore, $M_K = 12, k = 1, 2$. The preprocessing module assigns a severity level to these alerts based on their relation to the stages of the multi-stage attacks. In the case of more than one alert which leads to a state, the higher severity level is given to the alert which indicates that the attack is progressing. The training of the two HMMs is conducted based on a five-state model ($N_k = 5, k = 1, 2$), which corresponds to the five stages in each attack. A discussion on the training and the parameterization of each HMM is given in Appendix A.

For the completeness of our evaluation, several scenarios of the two simultaneous attacks are generated with a varying degree of interleaving to test the performance of the proposed architectures. For some cases,
we compare the three architectures of Figs. 2, 3, and 4. The reason for using the generic architecture of Fig. 2 for the comparison is that no evaluation has been done in the literature for multiple multistage attacks. For all the results, the x-axis shows the running count of alerts as they are generated by Snort. For evaluation purposes, we propose three performance metrics, in addition to the widely used state probability metric [24]. The proposed metrics are: (1) the attack risk probability which provides insight to the speed of the attacks, (2) the detection error rate performance, which measures how much error is generated by an architecture in estimating states, and (3) the number of correctly detected stages. The justification of these metrics is given in the following subsections.

5.1 Generating Alert Interleaving Scenarios

Based on the two multi-stage attacks in the DARPA2000 dataset, we alter the timestamp of some of these alerts in both attacks so that we can generate a single sequence of alerts that is composed of a mix of the two attacks without altering the temporal order of the original alerts. In addition, the IP addresses of the hosts attacked by Attack 2 (LLDDOS 2.0.2) are also changed. The reason for this modification is to simulate two simultaneous attacks intruding into a network. Fig. 5 shows several scenarios of interleaved alerts for two simultaneous DDoS attacks. Note, the degree of interleaving increases from Scenario 1 to Scenario 4 indicating an increase in the sophistication of actions and complexity of attacks. Since Attack 2 takes a shorter time to compromise the target and launch DDoS, we manipulate the timestamps of Attack 2 so that it spreads across different times of Attack 1. Note also, in this experiment Case Study 1, we only implement systematic interleaving scenarios and no random interleaving scenario is used. The y-axis in Fig. 5 represents the alert severity based on the preprocessing module. Based on these scenarios the performance results of the proposed architectures are given in the following subsections.

5.2 Probability of State Estimation and the Effect of Interleaving

In this subsection, we present the state probability, \( \gamma_i(t) \), from (1) and (5) for \( i = 1, \ldots, 5 \) with two observation lengths, \( T = 10 \) and \( T = 500 \). Regarding \( T = 500 \), it can be seen from Figs. 6 and 8 that Architecture I can estimate the states of both attacks with a high probability, especially for States 1, 2, 4, and 5. However, as the degree of interleaving increases from Scenario 1 to Scenario 4, Architecture I fails to detect many states. For example, for Scenario 3, States 3 and 4 of Attack 2 are not detected, as depicted in Fig. 6c. For Scenario 4, Architecture I performs very poorly as it fails to detect all the states of both attacks, as depicted in Fig. 6d. For \( T = 10 \), Architecture I shows a small improvement in detecting States 3 and 4 for Scenarios 1 and 3, respectively, as can be seen from Fig. 6c and Fig. 8. The reason for the poor performance of Architecture I is that the increasing degree of interleaving between alerts allows for more interfering alerts to exist within a given sequence. These conditions cause the Viterbi algorithm to incorrectly determine the state probability of the non-interfering alerts.

Architecture II, on the other hand, performs better as compared to Architecture I in terms of estimating correct states of all incoming alerts for both \( T = 10 \) and \( T = 500 \). This performance improvement, even for higher degrees of interleaving, can be observed from Figs. 6, 7, and 8. The reason behind this performance improvement for Architecture II is the presence of the demultiplexing module that helps each HMM to process only relevant attack alerts. Note, for both architectures the state probability for State 3 of Attack 1 is very low for both values of \( T \) because Snort does not produce enough alerts for this stage.

We observe discontinuity in the state probability plot of Architecture I in Figs. 6 and 8 a condition that results when both of the HMMs return to State 1 whenever interfering alerts exist from other attacks. However, in

1. Note: The terms detecting a state and estimating a state are used interchangeably in this paper.
Fig. 6: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 Based on Architecture I, T=500

Fig. 7: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 Based on Architecture II, T=500

Fig. 8: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 Based on Architecture I, T=10

Fig. 9: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 Based on Architecture II, T=10
Architecture II, as the alerts from different attacks are demultiplexed prior to their processing by the HMMs, the states of the HMMs are not interrupted. Fig. 10 shows the importance of considering $\epsilon_2$ in designing the HMM used by Architecture I. In this experiment, we choose $\epsilon_2 = 0.001$, which is a small value that does not significantly affect the transition probability matrices, $A_1$ and $A_2$, obtained from training. Note, $\epsilon_2 = 0$ represents the case of generic architecture, for which returning to State 1 is not allowed when HMM1 receives alerts belonging to Attack 2 or when HMM2 receives alerts belonging to Attack 1. Setting $\epsilon_2 = 0$ reduces the accuracy of state detection for the two attacks, as can be seen in Fig. 10. For example, for interleaving Scenario 2, Fig. 10b provides no estimate for state probability of States 2 and 4 for the first 200 alerts when $\epsilon_2 = 0$, as compared to Fig. 10d when $\epsilon_2 = 0.001$. A similar observation can be made by comparing Figs. 10c and 10d for the first 350 alerts of State 2.

In summary, Figs. 6b, 7b, 8b, and 9b show that no significant change occurs in the state detection performance of each architecture as the observation length changes from 10 to 500 alerts. Moreover, Architecture II is more robust in terms of having a better state probability estimation metric at a higher degree of interleaving as compared to Architecture I.

5.3 Attack Risk Probability

We define the first proposed performance metric as the attack risk probability, which is the probability of how far an attack is from compromising the target, i.e., reaching the final state. The calculation of this attack probability depends on the estimated state probability ($\gamma_i(t)$) averaged over the total number of states. Its value gets updated at every observation length in a non-decreasing manner, as shown below in (6):

$$P_{\text{attack}}_k(t) = \frac{\sum_{i=1}^{N_k} \gamma_i(t) s_k}{N_k}$$

$$t = 1, \ldots, T \quad i = 1, \ldots, N_k \quad k = 1, \ldots, 2$$

This performance measure can help in tracking the progress of each attack, especially when there are multiple organized attacks. It can be noted that, the rate at which the attack risk probability changes with respect to alerts gives an indication of how fast or slow an attack is progressing. Consequently, this measure can help in prioritizing response actions for each ongoing attack.

Figs. 11 and 12 show the attack risk probability for both DARPA multi-stage attacks using Architecture I and II for different interleaving scenarios and for the two observation lengths, $T = 10$ and $T = 500$. The results shown are for Scenarios 3 and 4, as they are relatively more complex to detect. Fig. 12 shows that Architecture II can track the progress of Attacks 1 and 2 for both interleaving scenarios accurately based on the knowledge of the generated input alerts shown in Figs. 5 and 9. Note, there is no significant difference is shown between the case of $T = 10$, and $T = 500$. Also note, after 100 alerts, Attack 2 progresses relatively quickly, and reaches the compromised state well before Attack 1. In contrast, however, Architecture I underestimates the progress of Attacks 1 and 2, as shown in Figs. 11 and 12, because Architecture I fails to detect some of the states for Scenarios 3 and 4, as illustrated in the previous subsection. Fig. 11 shows that both attacks progress at a slow pace. However, this discrepancy is not true as depicted in Fig. 12, where Architecture II shows both attacks progress quickly at different rates. For instance, Attack 2 reaches 80% after 100 alerts, while Attack 1 reaches 80% after 300 alerts. Moreover, Fig. 11 shows that Attack 1 progresses faster than Attack 2, which is also not true, as depicted in Fig. 12, which indicates Attack 2 is faster than Attack 1. This inaccurate detection of attacks can adversely affect response decisions, especially, when a priority-based mechanism is employed.

5.4 Error Rate Performance

The next performance measure we propose is the error rate ($ER$), which is the ratio of the number of errors resulting from the inconsistency between the type of an alert and the corresponding estimated state relative to the total number of incoming alerts. Formally, $ER$ is given by the following equation:

$$ER = \frac{\text{Number of incorrect detected states of the incoming Alerts}}{\text{Total number of Alerts}} \times 100$$

(a) SC2, $\epsilon_2 = 0$
(b) SC2, $\epsilon_2 = 0.001$
(c) SC3, $\epsilon_2 = 0$
(d) SC3, $\epsilon_2 = 0.001$

Fig. 10: Effect of $\epsilon_2$ on State Probability Based on Architecture I
Note, the exact state corresponding to every incoming alert is considered based on the knowledge of the input alerts and their corresponding states. The reasons for inconsistency between the type of alerts and their detected states are: (1) the presence of interfering alerts, and (2) the state estimation error resulting from the enforcement of the left-right HMM model along with some of the observations which may be out of sequence due to the packets generated by the attacker. In Subsection 5.6, we analyze the effect of False Positives (FPs) and False Negatives (FNs) introduced by the Snort alert generation system on the state estimation error.

Fig. 11 shows the plot of $ER$ for different interleaving scenarios and for several values of $T$. Note, the error for Architecture I is due to the aforementioned reasons (1) and (2), while the error for Architecture II, is due to only reason (2). It can be seen from the figure that the proposed architectures outperform generic architecture (Fig. 2) for interleaving Scenarios 1, 2, and 3. However, for Scenario 4, both Architecture I and the generic architecture have similar $ER$, which is higher than Architecture II. It can also be noted that the $ER$ for Architecture II remains almost constant with respect to $T$ and is also the same for all scenarios. Similarly, Architecture I has almost constant $ER$ with respect to $T$; however, its $ER$ performance gets worse as the degree of interleaving increases as compared to Architecture II. For instance, for Scenario 4, the $ER$ for Architecture I is as high as 77% as compared to Architecture II which has a value of 22%. Note, for the generic architecture, the $ER$ generally increases with $T$ and saturates to a value. The main reason for this trend is the same as aforementioned reason (1) and the lack of capability of this architecture to distinguish between alerts from two different attacks. In addition, due to the same reason, the $ER$ of the generic architecture also increases from Scenario 1 through Scenario 4.

5.5 Number of Correctly Detected Stages per Attack

The third performance measure we propose is the number of correctly detected stages per attack, which allows us to analyze the security impact due to missing or incorrectly detecting stages in a multi-stage attack, especially in consideration of response actions. We compare between architectures in terms of the number of detected stages per attack.

This measure is computed as follows. As we know the correspondence between alerts and stages (or states) of the attacks based on the knowledge of the DARPA2000 dataset, we compare the estimated states from each HMM with the exact states. Table 3 provides the results for three different values of $T$. It can be observed that Architecture II outperforms Architecture I in correctly detecting more stages for both attacks. The performance of the two architectures is the same for the interleaving Scenario 2, as both of them can detect stages 1, 2, 4, and 5 but not 3. This can be seen in Figs. 6, 7, 8, and 9. For Scenarios 3 and 4, Architecture II detects more
Fig. 13: State Detection Error Rate at Various interleaving Scenarios

Fig. 14: Comparison between Architectures I and II in detecting stages of Attack 2 for Scenario 3

Fig. 15: The Impact of False Positives on the State Detection Error Rate of Architectures I & II in Scenarios 1-4 Using Various False Discovery Rates ($FDR = 0\% - 50\%$) - The Observation Window Size = 500 and the Number of Experiments = 100

Fig. 16: The Impact of False Negatives on the State Detection Error Rate of Architectures I & II in Scenarios 1-4 Using Various False Negative Rates ($FNR = 0\% - 50\%$) - The Observation Window Size = 500 and the Number of Experiments = 100
TABLE 3: Number of Correctly Detected Stages per Attack at Various Interleaving Scenarios

| Interleaving Scenario | Architecture | Attack | T = 10 | T = 100 | T = 500 |
|-----------------------|--------------|--------|--------|---------|---------|
| Scenario 2            | I            | Attack 1 | 4      | 4       | 4       |
|                       |              | Attack 2 | 5      | 5       | 5       |
|                       | II           | Attack 1 | 4      | 4       | 4       |
|                       |              | Attack 2 | 5      | 5       | 5       |
| Scenario 3            | I            | Attack 1 | 4      | 4       | 5       |
|                       |              | Attack 2 | 4      | 4       | 3       |
|                       | II           | Attack 1 | 4      | 5       | 5       |
|                       |              | Attack 2 | 5      | 5       | 5       |
| Scenario 4            | I            | Attack 1 | 4      | 4       | 5       |
|                       |              | Attack 2 | 5      | 5       | 5       |
|                       | II           | Attack 1 | 4      | 5       | 5       |
|                       |              | Attack 2 | 5      | 5       | 5       |

5.6 Impact of False Positives (FPs) and False Negatives (FNs)

The security alerts generated by an IDS are, in general, noisy and suffer from both FPs and FNs. In the former case, the IDS (e.g., Snort) generates false alerts when no attack attempts are happening in the network, and in the latter case, the IDS fails to detect exploit attempts and does not generate alerts [40]. In our evaluation of the proposed architectures, similar effects are observed, as discussed below.

We have conducted several experiments to study the impact of FPs and FNs for the proposed HMM architectures. In our experiments, we synthesize the dataset by eliminating some of the True Positive alerts (TPs), in order to mimic FNs, and inject some FPs into the observation sequence in a randomized fashion. In our experiments, we vary the False Discovery Rate (FDR) and the False Negative Rate (FNR) from 0% to 50% for the alert generation system (Snort). In addition, due to randomized injection and elimination, we conduct 100 experiments for each interleaving scenario and for each value of FDR and FNR to identify any potential outliers. Note, in our experiments, we assume that the FP error is uniformly induced by all of the alert generation rules employed by Snort. In other words, the effect of the FPs is uniformly distributed over generated alerts by Snort.

The results for the impact of FDR for both architectures are shown in Fig. 15. It can be noticed that the performance of Architecture II degrades with the increase of FDR. This lowered performance is expected as some of FPs are also “demultiplexed” and affect the TPs in their respective substreams when each of these substreams is processed by the associated HMM template.

However, for Architecture I, the general trend observed is an improvement in the detection error rate performance which is more noticeable for Scenario 4 (Fig. 15d). A plausible explanation for this trend is that the FPs in the whole stream either maintain the current state or allow for a transition to a subsequent state in the HMM. The DARPA2000 dataset contains a high number of observations related to State 1 as compared to other states. Therefore, under the assumption of a uniform injection of FPs in the alert stream, the likelihood of the HMM staying in State 1 increases with the increase in FDR. Note, an HMM template for Architecture I always leads to State 1 for unrelated observations. Therefore, as the percentage of FDR increases, this tendency of staying in State 1 also increases with the high degree of interleaving due to the increase in the number of unrelated observations.

The effect of FNs of the IDS system (Snort) on both architectures is shown in Fig. 16. The effect of the FNR on the performance of Architecture II is shown in Fig. 16. Note, some of the TPs which are eliminated by FNs may belong to the erratic behavior of the attacker (which is reason (2) mentioned in Subsection 5.4), while some other TPs eliminated by FNs are legitimate (i.e., correctly sequenced) alerts. A positive effect is shown on performance in the case of erratic behavior results in improved performance of detection error rate, while for the case of legitimate alerts, the performance degrades. We can notice such improvement and degradation in the performance for different values of FDR and FNR as shown in Figs. 15 and 16.

For Architecture I, in addition to reason (2), the reason (1) (mentioned in Subsection 5.4) also comes into play, whereby FNs can eliminate some unrelated alerts and...
Fig. 17: State Probability of Synthesized Attacks 1 and 2 for the Interleaving Scenario 3 Detected by HMM1 and HMM2 Based on both Architectures, T=500

thereby reduce the possibility of transition to State 1 and increasing the possibility of transition from a given state to the next state. This improvement in the performance is more noticeable for Scenario 4 where the prospects of making such forward transitions are higher.

5.7 Performance Evaluation Using Synthesized Datasets - Case Study 2

The evaluation experiments in the previous case study (Subsections 5.1-5.6) have been implemented with DARPA2000 simultaneously interleaved attacks. Due to the limitation of this dataset in terms of number of scenarios and due to the lack of availability of datasets with a large number of attacks, in this case study, we generate synthesized datasets that contain different instances of the DARPA2000 multi-stage attacks. Note, it is important to point out that our objective is not to evaluate a specific dataset, but rather to evaluate the proposed architectures for detecting complex multi-stage attacks that are orchestrated by an adversary through interleaving. The design of the architectures is generic in the sense they can process any dataset with multiple attacks. Specifically, the goal of this case study is to study the effectiveness of the proposed architectures when tested on various datasets that have multi-stage attack instances which vary from the trained HMM templates. The importance of this evaluation is that, in reality, the attacker(s) may not follow the exact same steps for the same multi-stage attack type, for example, in terms of the targeted nodes or the number of attempts.

In particular, an HMM generator [42], which generates sequences for HMM, is used to orchestrate several instances of a multi-stage attack type. In this case study, the original DARPA2000 dataset is used for training, and the generated synthesized datasets is used for testing.

5.7.1 Performance Evaluation - Two Synthesized Multi-stage Attacks

Fig. [17] shows the state probability of synthesized Attacks 1 and 2 for the interleaving Scenario 3 detected by HMM1 and HMM2 for Architecture I, Fig. [17b], and for Architecture II, (Fig. [17c]). It can be observed in Fig. [17] that the results of the case study of synthesized multi-stage attacks are consistent with the previous results for Scenario 3 from Case study 1, discussed in Subsection 5.2 (Figs. [5], [6], [7]), in terms of detection performance and state estimation. In particular, Architecture II detects all stages of the interleaved multi-stage attack scenario. In contrast, Architecture I fails to detect State 5 of Attack 2 as it estimated the state as State 2 and State 3 due to the noisy observations resulting from unrelated alerts.

5.7.2 Performance Evaluation - Four Synthesized Multi-stage Attacks

The proposed architectures can be applied to more than two simultaneous attacks, as well as attacks with different instances. We have conducted several experiments, using synthetic datasets, to study the effect of having more than two simultaneous multi-stage attacks on the performance of the proposed architectures. In particular, Architecture II performs better than Architecture I since it depends essentially on the demultiplexing operation. However, with more than two multi-stage attacks, more computations are involved, especially in the demultiplexing module and also in the HMM database component. Consequently, the mean time to demultiplex the stream and to estimate the state for a window size of 100 increases from 0.46 milliseconds to 1.9 milliseconds. Architecture I works well with more than two attacks, but its detection performance deteriorates significantly with a large number of attacks and a higher degree of interleaving. Fig. [18] shows the results for a scenario of four multi-stage attacks. In this scenario, the attacker(s) in Attack 3 and Attack 4 attempt to hide an attack with some of their previous attempts that he/she exploited successfully, which represent two quite different instances from the trained HMM templates for these attacks. Although the sequence of observations has unrelated observations from four different multi-stage attacks, especially in the window between 300 and 400, Fig. [18] shows that the proposed architecture estimates the progress of the attack correctly. Due to page limit, we show the results for only one scenario of interleaving from the four multi-stage attacks.
6 CONCLUSION

This paper addresses the detection problem of interleaved multiple multi-stage attacks intruding into a computer network. We emphasize the importance of this problem by showing how interleaving and stealthy attacks can deceive the detection system. Therefore, we propose two architectures based on a well-known machine learning technique, i.e., the Hidden Markov Model, and provide their performance results and computational complexity. Both architectures can track interleaved attacks by detecting the correct states of the system for each incoming alert. However, as the degree of interleaving among attacks increases, Architecture II, which employs a demultiplexing mechanism, exhibits more robustness and better performance as compared to Architecture I. For the performance assessment of these architectures, we propose three performance metrics which include (1) attack risk probability, (2) detection error rate, and (3) the number of correctly detected stages. The DARPA2000 dataset is chosen to synthesize interleaved multi-stage attack scenarios and to demonstrate the efficacy of the proposed architectures. The proposed architectures are generic in terms of their capability to process any dataset that contains multiple interleaved multi-stage attacks.

7 ACKNOWLEDGMENT

This research was supported by the grants from Northrop Grumman Corporation and US National Science Foundation (NSF) Grant IIS-0964639 and was supported by King Abdulaziz University. Distribution Statement A: Approved for Public Release; Distribution is Unlimited #18-1658, Dated 10/30/18.

REFERENCES

[1] CNET, “Wannacry ransomware: Everything you need to know,” 2017, accessed: 2017-10-22.
[2] Washingtonpost, “Massive cyberattack hits europe with widespread ransom demands,” 2017, accessed: 2017-10-22.
[3] S. Jajodia and M. Albanese, “An Integrated Framework for Cyber Situation Awareness”. Springer International Publishing, 2017.
[4] N. Lukitarhan, X. Jia, L. Hu, and N. Xie, “Multi-stage attack detection algorithm based on hidden markov model,” in International Conference on Web Information Systems and Mining. Springer, 2012.
[5] J. Navarro, A. Deruyver, and P. Parreng, “A systematic survey on multi-step attack detection,” Computers & Security, 2018.
[6] X. Qin and W. Lee, “Attack plan recognition and prediction using causal networks,” in 20th Annual Computer Security Applications Conference, 2004, pp. 370–379.
[7] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, “Comprehensive approach to intrusion detection alert correlation,” IEEE Transactions on Dependable and Secure Computing, 2004.
[8] B. C. Cheng, G. T. Liao, C. C. Huang, and M. T. Yu, “A novel probabilistic matching algorithm for multi-stage attack forecasts,” IEEE Journal on Selected Areas in Communications, 2011.
[9] L. Wang, A. Liu, and S. Jajodia, “Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts,” Computer Communications, vol. 29, no. 15, pp. 2917 – 2933, 2006.
[10] D. S. Fava, S. R. Byers, and S. J. Yang, “Projecting cyberattacks through variable-length Markov models,” IEEE Transactions on Information Forensics and Security, vol. 3, no. 3, pp. 359–369, 2008.
[11] H. Du, D. F. Liu, J. Holsopple, and S. J. Yang, “Toward ensemble characterization and projection of multistage cyber attacks,” Proceedings - International Conference on Computer Communications and Networks, ICCCN, 2010.
[12] A. A. Ramaki, M. Amini, and R. Ebrahimi Atani, “RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection,” Computers and Security, vol. 49, pp. 206–219, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2014.10.006
[13] F. Manganelli, M. Marchetti, and M. Colajanni, “Multistage attack detection and alert correlation in intrusion detection systems,” in Information Security and Assurance, T.-h. Kim, H. Adeli, R. J. Robles, and M. Balitanas, Eds., 2011.
[14] P Ning and D Xu, “Learning attack strategies from intrusion alerts,” in Proceedings of the 10th ACM Conference on Computer and Communications Security, ser. CCS ’03, New York, NY, USA: ACM, 2003, pp. 200–209. [Online]. Available: http://doi.acm.org/10.1145/948108.948137
[15] L Huaying, W Wu, W Ruimei et al., “A real-time network threat recognition and assessment method based on association analysis of time and space,” Journal of Computer Research and Development, vol. 51, no. 5, pp. 1039–1049, 2014.
[16] C. Onwubiko, “Situational Awareness in Computer Network Defense: Principles, Methods and Applications: Principles, Methods and Applications”. IGI Global, 2012.
[17] Z. Zali, M. R. Hashemi, and H. Saidi, “Real-time attack scenario detection via intrusion detection alert correlation,” in Information Security and Cryptology (ISISC), 2012 9th International ISC Conference on. IEEE, 2012, pp. 95–102.
[18] F. Alserhani, M. Aklilu, I. U. Awan, A. J. Cullen, and P. Mirchandani, “Mars: multi-stage attack recognition system,” in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010, pp. 753–759.
[19] W. J. Zhang Hengwei, Yang Haopu and L. Tao, “Multi-step attack-oriented assessment of network security situation,” International Journal of Security and Its Application, 2017.
[20] D. Ourston, S. Matzner, W. Stump, and B. Hopkins, “Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks,” Proceedings of the 36th Annual Hawaii International Conference on System Sciences, p. 334, 2003.
[21] A. Årnes, K. Sallhammer, K. Haslum, T. Brekne, M. E. G. Moe, and
S. J. Knapskog, “Real-time risk assessment with network sensors and intrusion detection systems,” in International Conference on Computational and Information Science. Springer, 2005.

K. Haslam, A. Abraham, and S. Knapskog, “Fuzzy online risk assessment for distributed intrusion prediction and prevention systems,” Proceedings - UKSim 10th International Conference on Computer Modelling and Simulation, EURONSIM/UKSim2008, pp. 216–223, 2008.

H. Farhadi, M. Amirihaeri, and M. Khansari, “Alert Correlation and Prediction Using Data Mining and HMM,” The ISC Int’l Journal of Information Security, vol. 3, pp. 77–101, 2011.

A. S. Sendi, M. Dagenais, M. Jabbarifar, and M. Couture, “Real time intrusion prediction based on optimized alerts with hidden markov model,” JNW, vol. 7, no. 2, pp. 311–321, 2012.

S. Zonouz, K. M. Rogers, R. Berthier, R. B. Bobba, W. H. Sanders, and T. J. Overbye, “Scpsc: Security-oriented cyber-physical state estimation for power grid critical infrastructures,” IEEE Transactions on Smart Grid, vol. 3, no. 4, pp. 1790–1799, 2012.

H. A. Kholidy, A. Erradi, S. Abdelwahed, and A. Azab, “A finite state hidden markov model for predicting multistage attacks in cloud systems,” in Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on, 2014.

P. Holgado, V. A. Villagra, and L. Vazquez, “Real-time multistep attack prediction based on hidden markov models,” IEEE Transactions on Dependable and Secure Computing, 2017.

A. R. Ali, R. Abbas, and J. J. Abbas, “A systematic review on intrusion detection based on the hidden markov model,” Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 11, no. 3, pp. 111–134, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11377

M. Albanese, S. Jajodia, A. Pugliese, and V. S. Subrahmanian, “Scalable analysis of attack scenarios,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6879 LNCS, pp. 416–433, 2011.

L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

L. L. M. I. of Technology, “Defense advanced research projects agency dataset (darpa),” 2015.

J. Yin and Y. Meng, “Abnormal behavior recognition using self-adaptive hidden markov models,” in Image Analysis and Recognition, M. Kamel and A. Campilho, Eds. Springer Berlin Heidelberg, 2009.

Jason Kubes works as a principal cyber architect and research scientist in Washington, DC for Northrop Grumman Corporation. He has over 20 years of experience concentrated in information systems design analytics, business/mission security architecture, enterprise risk management, information assurance research, and business consulting. He has an M.S. in information assurance (MSIA) and a B.S. in computer science from Iowa State University.

Tawfeeq Shawly received the M.S. and Ph.D. degrees in communications, networking, and signal processing from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2016 and 2019, respectively. He is an Assistant Professor in the Electrical Engineering Department at the King Abdulaziz University in Rabigh, Saudi Arabia. His research interests include security of networked cyber-physical systems, artificial intelligence and machine learning.

Ali Elghariani (S’12-M’14) received the B.S. and M.S. degrees in electrical and electronic engineering from the University of Tripoli, Tripoli, Libya. He obtained his Ph.D. degree in communications, networking, and signal processing from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2014. Currently, he is a research engineer at XCOM-Labs in San Diego CA. When this paper was submitted, he was a visiting researcher at Purdue University, West Lafayette IN. In 2015-2016, he was a Lecturer at the Department of Electrical and Electronic Engineering, University of Tripoli, Tripoli, Libya. His research interests include signal detection and channel estimation in large-scale MIMO systems, mmwave communication, optimization techniques in wireless communications, and network security.

Arif Ghafoor is a professor in the School of Electrical and Computer Engineering at Purdue University. His research interests include multimedia information systems, database security, and distributed computing. He is a Fellow of the IEEE.