DESIGN AND EVALUATION OF HIDDEN MARKOV MODEL BASED ARCHITECTURES FOR DETECTION OF INTERLEAVED MULTI-STAGE NETWORK ATTACKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Tawfeeq A. Shawly

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Arif Ghafoor, Chair

School of Electrical and Computer Engineering

Dr. Walid G. Aref

Department of Computer Science

Dr. Elisa Bertino

Department of Computer Science

Dr. Jianghai Hu

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School Graduate Program

This thesis is dedicated to my mother, to my wife and to my sons.

ACKNOWLEDGMENTS

I would like to sincerely thank Professor Arif Ghafoor for his advice, guidance, and support during my graduate study at Purdue University. I would also like to express my gratitude to Professor Walid Aref, Professor Elisa Bertino, and Professor Jianghai Hu for their guidance and participation as members of my PhD committee. I would also like to acknowledge the support provided by King Abdulaziz University in Saudi Arabia during my graduate studies.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

λ_k	The HMM model for Attack k
S	The set of attack states
O	The observation sequence of IDS alerts
N	The number of states of the system
M	The number of distinct observation symbols
A	The state transition probability, N x N matrix
B	The observation emission probability, N x M matrix
π	The initial states probabilities of the HMM

ABBREVIATIONS

ATM	Adaptive Threat Management
CBS	Cyber-based Systems
HMM	Hidden Markov Model
IDS	Intrusion Detection System
ML	Machine Learning
MSA	Multi-stage Attacks

ABSTRACT

Shawly, Tawfeeq A. PhD, Purdue University, December 2019. Design and Evaluation of Hidden Markov Model Based Architectures for Detection of Interleaved Multi-stage Network Attacks. Major Professor: Arif Ghafoor.

Nowadays, the pace of coordinated cyber security crimes has become drastically more rapid, and network attacks have become more advanced and diversified. The explosive growth of network security threats poses serious challenges for building secure Cyber-based Systems (CBS). Existing studies have addressed a breadth of challenges related to detecting network attacks. However, there is still a lack of studies on the detection of sophisticated Multi-stage Attacks (MSAs).

The objective of this dissertation is to address the challenges of modeling and detecting sophisticated network attacks, such as multiple interleaved MSAs. We present the interleaving concept and investigate how interleaving multiple MSAs can deceive intrusion detection systems. Using one of the important statistical machine learning (ML) techniques, Hidden Markov Models (HMM), we develop three 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 set of HMM templates of known attacks and exhibit varying performance and complexity.

For performance evaluation, 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 conducted to demonstrate the efficacy of the proposed architecture in the presence of multiple multi-stage attack scenarios, and in the presence of false alerts with various rates.

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. Cyber attacks can affect and downgrade functionalities and missions of critical infrastructures [1]. For instance, in December 2015, hackers were able to successfully compromise information systems of three energy distribution companies in Ukraine and temporarily disrupt the power supply to consumers [2]. Another recent incident in May 2017 occurred when the "WannaCry" ransomware attack was detected after it locked up over 200,000 servers in more than 150 countries [3]. 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 [4].

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 [5].

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 (MSAs), and therefore are hard to predict [6]. 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 [7].

One example of multi-stage attacks is the Distributed Denial-of-Service attack (DDoS), which consists of the following main stages. At the beginning, the attacker tries to identify potential vulnerabilities by scanning the targeted network. Then, he attempts to break into vulnerable hosts and to compromise them. After that, the attacker installs malware in the compromised hosts, and eventually initiates a DDoS attack to the targeted server, which is accessible from all the exploited hosts [8].

The detection of multi-stage attacks is a challenge for existing threat detection techniques. Furthermore, launching multiple such attacks simultaneously in the network, in order to stealth certain attacks within others, exacerbates this challenge [7]. The main challenges for detecting interleaved multi-stage attacks include: modeling multi-stage attacks in terms of security states and in the presence of mixed observations (IDS alerts); designing an efficient architecture that detects and tracks the progress of interleaved multi-stage attack in the presence of IDS false alarms; and evaluating the detection performance of the architecture, given that there is a lack of standard procedures and a lack of availability of datasets with interleaved multi-stage attack scenarios.

In this dissertation, we address the challenges of detecting interleaved multi-stage attacks. In particular, using one of the important ML techniques, Hidden Markov Models (HMM), we propose three Intrusion Detection System (IDS) 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. In particular, the proposed architectures can identify, at any point in time, how slowly or quickly each attack is progressing; and which security state each attack is in. The design of these architectures relies on modifying HMM model parameters to detect multiple multi-stage attacks in the presence of interleaved alerts.

The proposed architectures utilize an IDS (e.g., SNORT [9]) to monitor the network traffic and to raise alerts for any suspicious activities. We quantitatively evaluate the detection accuracy of architectures using multiple simulated multi-stage attack scenarios where we control the behavior of SNORT to study the effect of false and missing alerts. In other words, we vary the behavior of the attacker, in terms of stealthiness, and vary the behavior of the IDS, in terms of false and missing alarms, to demonstrate the efficacy of the proposed architectures.

1.1 Research Motivation

One of the most essential requirements for having high-assurance Cyber systems is to develop advanced and sophisticated attack detection and prediction systems [5]. Most detection systems have the capability to detect a single-stage attack or to detect each of the stages of a multi-stage attack independently. However, due to the inability to analyze the chain of the attack activity as a whole, the detection of multi-stage attacks poses a daunting challenge to the existing intrusion detection methods.

This challenge is exacerbated if multiple multi-stage attacks are interleaved intentionally by the attacker(s). The difficulty in detecting interleaved multi-stage attacks comes from unrelated observations (i.e., interfering alerts) resulting from unrelated attacks that conceal the details of the activity chains of multi-stage attacks. These unrelated observations cause an interference for the multi-stage detection system which reduces the certainty about the current situation. Furthermore, the stream of observations might contain a large number of false alerts or incomplete observations [10]. As a result, this noisy stream of observation can degrade the performance of the multi-stage detection system.

To elaborate on this challenge, Fig. 1.1 exhibits three multi-stage attacks interleaved in five possible scenarios. The correspondence between the Alert type and stages for some observations of these multi-stage attacks is shown in the top-right table. A hypothetical real-time state estimation of Attack 1 for scenarios (3), (4)

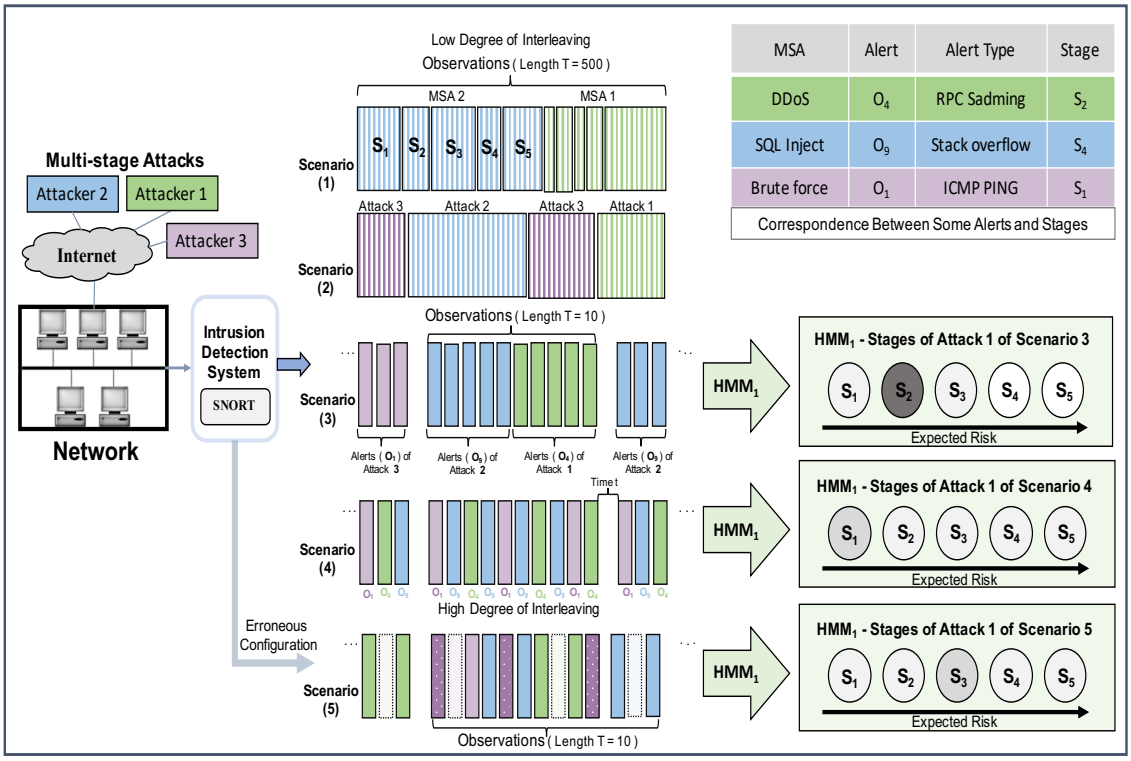


Fig. 1.1.: State Estimation of Interleaved Multi-stage Attacks Scenarios at Time t

and (5), assuming that Attack 1 has five stages, is shown on the right side of the figure. A darker color represents a higher probability for the HMM state estimation (i.e., indicates a higher degree of certainty about the current state). Fig. 1.1 shows an example of the impact of interleaving among alerts of three multi-stage attacks on the performance of an HMM-based state estimation. For instance, the degree of interleaving in scenario (3) in Fig. 1.1 is lower than in scenario (4), and therefore, at time t, the certainty about the current state for Attack 1 in scenario (3) can be higher than in scenario (4) (which is supposed to be State 2 for both scenarios). Moreover, in scenario (5), we assume an erroneous configuration for the IDS that can cause a high rate of false and missing alarms. Hence, due to both reasons, the interleaving of multi-stage attacks and the erroneous configuration of the IDS, the uncertainty about the current state for Attack 1 in scenario (5) can also be higher than in scenario (3).

1.2 Summary of Challenges and Contributions

This research addresses several challenges to the detection of multiple multi-stage attacks which include:

1. Multiple MSAs challenges (e.g., Scenario (3) in Fig. 1.1):
    - How to model each multi-stage attack in terms of HMM states?
    - How to design an efficient architecture that can track the progress of multiple attacks?
    - How to deal with unrelated and shared observations?
2. Highly Interleaved MSAs challenges (e.g., Scenario (4) in Fig. 1.1):
    - How to detect a multi-stage attack when an attacker(s) performs highly interleaved multi-stage attacks with the intention to hide an attack (i.e. stealthy attacks)?
    - How to eliminate (or reduce) the interference using a demultiplexing approach?
3. Erroneous and incomplete observations (e.g., Scenario (5) in Fig. 1.1):
    - How to reduce the noise due to erroneous IDS ("de-noising")?
    - How to detect multi-stage attacks with incomplete information (missing alerts)?
4. Performance Evaluation
    - How to efficiently evaluate the performance (metrics and datasets)?

      - The development of an approach to accurately quantify and measure the detection performance of such an architecture.

      - Since no standard public dataset is available that can provide interleaved traffic from simultaneous multiple attacks, the generation of this type of dataset poses a challenge to the research community,

To address the above challenges, we propose in this thesis three architectures based on HMM formalism. The proposed architectures exhibit varying detection performance and processing complexities. These 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 R-LRHMM 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 [5, 11].

Specifically, in Chapter 3, we present the first proposed architecture (i.e., Resetting Left-Right HMM (R-LRHMM)) and the proposed performance evaluation metrics. The design of R-LRHMM relies on modifying HMM model parameters to detect multiple multi-stage attacks in the presence of mixed alerts using a reset approach. In particular, as the rate of the unrelated observations from an unrelated attack(s) increases, the tendency of resetting the HMM model for the corresponding original attack to the initial state (State 1) and staying there also increases due to the increase in the number of unrelated observations. R-LRHMM can detect the multi-stage attacks in Scenario (1), (2) and (3) shown in Fig. 1.1 with a high detection performance and low detection delay as well, as discussed in Chapter 3. However, R-LRHMM fails to detect the stage of Scenario (4) and (5) due to the high degree of interference and the noisy observations in these two scenarios.

In Chapter 4, we introduce in more details the interleaving concept and investigate how interleaving multiple MSAs can deceive the HMM-based intrusion detection systems. Moreover, we propose the second architecture (i.e., Left-Right HMM Architecture with Attack Demultiplexer (LRHMM+AD)) and compare its performance and complexity with R-LRHMM. The design of LRHMM+AD relies on de-interleaving mixed alerts from different attacks prior to the HMM processing subsystem by using a Demultiplexer component. Further, we study the impact of false and missing alerts on the performance of both architectures in order to motivate the need for the third

proposed architecture (i.e., Highly Resilient Ergodic HMM (HR-EHMM)) which is presented in Chapter 5. LRHMM+AD can detect the multi-stage attacks in Scenario (1), (2), (3) and (4) shown in Fig. 1.1 with a high detection performance and with a higher complexity than in R-LRHMM. The design of HR-EHMM relies on modifying HMM model parameters to detect interleaved multi-stage attacks in the presence of false and missing alarms and hence by using a nullifying and a backtracking approaches, HR-EHMM can detect all the multi-stage attack scenarios shown in Fig. 1.1 as shown in Chapter 5.

Note, the reason for discussing R-LRHMM in this thesis, despite its low detection performance with some specific multi-stage attack scenarios, is to provide the tradeoff between the low computation (i.e., faster output from the system) and the performance cost in terms of the high state estimation error rate. On the other hand, LRHMM+AD and HR-EHMM yields to a better performance in terms of state estimation error rate but at the cost of high complexity incurred by introducing the demultiplexer in both architectures. In this thesis, we compare the three architectures in terms of their detection performance and design complexity.

1.3 Organization

The thesis is organized as follows. In Chapter 2 the background has been introduced along with the related work. In Chapter 3, an HMM-based threat detection and prediction mechanism is proposed (R-LRHMM). In Chapter 4, LRHMM+AD is introduced and its performance is compared with R-LRHMM. In Chapter 5, HR-EHMM is proposed. Chapter 6 concludes the thesis with plans for the future work.

2. BACKGROUND AND RELATED WORK

This chapter summarizes the research efforts regarding intrusion detection and prediction methods and provides some background material.

With the growing cyber threats, making the right decisions by security analysts has become an obstacle and the need to develop high assurance and resilience cyber-based systems is becoming increasingly important. Note that the term "resilience" refers to the ability to adapt to changing conditions and rapidly recover from disruption due to emergencies such as cyber attacks and in timely manner [12]. Existing studies have addressed a breadth of challenges related to detecting network attacks.

In order to make better-informed decisions, to take a proper response to a cyber Multi-stage Attack (MSA), and to keep the system's resilience at a high level under a progressing MSA, security analysts need to be aware of the current situation, the impact and evolution of an attack, the behavior of the attackers, the quality of available information and models, and the expected future of the current situation. Recently, researchers have shown an increased interest in investigating a breadth of challenges and research questions related to the design and development of resilience systems. Some of the fundamental questions that an effective cyber situation awareness system must be able to help answer are [5]:

1. Current situation:

    - What is the stage or the state of the ongoing MSAs?
    - What is the attacker's location?

    These questions aim to evaluate the effectiveness of the deployed Intrusion Detection System (IDS), and identifying the compromised and damaged assets.

Data gathered by security monitoring systems are used to answer such questions [1], [11], [13].

2. Evolution:

   - How is the MSA evolving?
   - What is the speed of the MSA propagation?
   - Can we track all the steps of an MSA?

   Such questions aim to evaluate the effectiveness of security monitoring systems facing ongoing attacks, once some stages of such attacks have been detected. Ideally, the output of the ATM system should be clear information of how the attack is progressing [14], [15], [16], [17].

3. Behavior:

   - How are the attackers expected to behave?
   - What are the strategies of attackers?
   - Can we evaluate the capabilities of attackers?

   These questions aim to evaluate the capability of modeling the attacker's behavior in order to understand his goals and strategies. In this case, the output of the ATM system is supposed to be a set of formal models of the attacker's behavior (e.g., stochastic models or game theoretic models) which takes into account the dynamicity of the attacker's behavior over time [18], [19].

4. Forensics:

   - How did the attacker reach the current situation?
   - What was he trying to achieve?

   These questions aim to evaluate the capability of analyzing the IDS logs and correlating observations to analyze how an attack originated and evolved so far.

Based on this information, security administrators can harden system configurations to eliminate any similar incidents in the future [20], [21].

5. Prediction:

   - Can we predict a plausible future of the current situation?

   These questions aim to evaluate the capability of predicting the attacker's possible moves in the future. In this case, the output of the ATM system is supposed to be a set of all possible alternative scenarios ranked probabilistically [22] [23] [24] [25].

6. Impact:

   - How is the attack impacting the system's functionality and missions?
   - Can we assess the risk and the damage of the multi-stage attack (past and future stages)?

   Answering this set of questions implies the capability of assessing the impact of an ongoing MSA accurately. In this case, the ATM system needs information about of the CBS assets and their dependencies, importance and criticality for the system's functionality and missions [26], [27].

7. Response:

   - What are the possible responses to an ongoing MSA(s) that keeps the system in high assurance and resilience levels?
   - Can we rank possible responses in terms of cost, system damage, recovery speed, or any other metric?
   - Can we find the optimal responses if we have multiple objectives?

   These questions aim to evaluate the capability of the decision making and response systems. The input to the ATM system is the situation awareness obtained in response to the previous questions. Ideally, the output is supposed

to be a set of all possible responses ranked based on assurance and resilience requirements [28], [29], [30], [31], [32].

There is a strict correlation and dependency between answers of many of these questions. For instance, the capability of predicting plausible moves of an attacker depends on the capability of modeling the attacker's behavior. Another example is that the capability of finding and ranking all possible responses depends, for example, on the capability of predicting possible future moves of an attacker and his speed, and the capability of assessing the impact of an ongoing attack accurately. The most challenging issues that affect almost all aspects of any cyber security framework are real-time requirements and scalability (e.g., in terms of estimating the overall security state of CBS and responding to ongoing multi-stage attacks in timely manner). Given the volumes of data involved in answering all these questions, the large number of infrastructure components that the framework deals with, and timing constraints as there are detection and assessment latencies, we need to define approaches that are not only effective and computationally efficient, but also respond to ongoing attacks optimally and in timely manner. This thesis aims to address many of these research questions and challenges as will be discussed later.

2.1 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 [33–35] or machine learning (ML) based techniques. Examples of ML techniques include Hidden Markov Models, Bayesian Networks, Clustering and Neural Networks [1, 24, 36, 37]. Correlation-based techniques, based on cause and effect relationships, mainly utilize attack-graphs when searching the possible stages of the attack [20, 38–42]. For example, the work in [38] focuses on the causal relationships between attack phases on the basis of security information. Onwubiko et al. [39] assesses network security through mining and restoring

the attack paths within an attack graph. A causal relations graph presented in [40], 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 [41, 42]. 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 to intrusion detection and the prediction of multi-stages attacks, [13, 22, 23, 43–51]. 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 [43]. 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. Holgado and her colleagues proposed a model to predict the progress of multi-step attacks by considering the hidden states as similar stages of a specific type of attack [49].

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.

2.2 Hidden Markov Models

Hidden Markov Model (HMM) is a well known and widely used sequential data modeling method which was introduced in the late 1960s [52]. Due to its rich math-

ematical structure, HMM is widely applied in real world applications such as speech recognition, handwriting pattern recognition, gesture recognition, and intrusion detection. An HMM is a doubly stochastic process with 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 [52]. The hidden process represents the states of the Markov chain (the actual stages of the attack). represents the observed alerts coming out from the IDS.

2.2.1 Hidden Markov Model Description

Consider that each attack is modeled using a distinct HMM model λ_k that encompasses all the following parameters,

$$\lambda_k = \{S_k, V_k, A_k, B_k, \pi_k\}, \;\; k = 1, 2, \ldots, K \tag{2.1}$$

where $S_k = \{s_{1_k}, \ldots, s_{n_k}, \ldots, s_{N_k}\}, s_{n_k}$ represents State n of Attack k, and N_k is the number of states in Attack k. Note, for multiple attacks, different attacks can have different values for the number of states (N). In this thesis, for simplicity of HMM computations, we assume that ongoing multi-stage attacks in the network can be modeled with the same number of security states. $V_k = \{v_{1_k}, \ldots, v_{m_k}, \ldots, v_{M_k}\}$ represents distinct observation symbols set for Attack k, where M_k is the total number of distinct observations for Attack k. In our model, we consider both the cases of $V_i \cap V_j = \phi$ and $V_i \cap V_j \neq \phi$, for $j \neq i$, and $j, i = 1, 2, \ldots, K$. That is, any two multi-stage attacks can have the same set of observations. A_k represents the state transition probability matrix of Attack k with dimension $N_k \times N_k$. Each element in this matrix, a_{ij}, represents the state transition probability form state i to state j as follows:

$$a_{ij} = P(x_{t+1} = s_j | x_t = s_i),$$
$$1 \leq i \leq N_k, \;\; 1 \leq j \leq N_k, \;\; x_t \in S_k$$

B_k is the observation emission probability matrix with dimension $N_k \times M_k$. The emission probability of the m^{th} observation of state j is represented by $b_j(m)$, as follows:

$$b_j(m) = Pr(o_t = v_m | x_t = s_j),$$
$$1 \leq j \leq N_k, \;\; 1 \leq m \leq M_k$$

Initial probability distribution vector (π) represents initial states probabilities of the HMM states with $\pi_i = Pr(x_1 = s_i)$, $1 \leq i \leq N$. The observation sequence (O) of length T for HMM is represented as $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$ where $o_t \in V_k$. As indicated in [52], HMM can deal with three main problems which are, the evaluation, hidden state decoding, and model training. Note, all of these problems need to be addressed for designing our architectures in order to determine and estimate the current belief state for a multi-stage attack. The three fundamental problems related to HMMs are as follows [52]:

1. The Evaluation: How to compute the probability of a sequence of observations given the model λ, $P(O|\lambda)$, using the Forward-Backward algorithm efficiently.

2. The Decoding: How to find the most likely states path given the model λ and the observation sequence $O = O_1, O_2, ..., O_t, ..., O_T$, using the Viterbi algorithm.

3. The Training: How to set and adjust the HMM parameters A, B, and π in order to maximize $P(O|\lambda)$ using the Baum-Welch algorithm.

The Baum-Welch algorithm is used first to perform offline training process related to the third problem, to find the best states sequence corresponding to the second problem, and to compute probabilities based on Viterbi and Forward-Backward algorithms for both, the first and second problems.

An important consideration that needs to be taken into account when choosing HMM for an application, is the type of the model. There are mainly two types of HMM in terms of state transition diagram. One is called ergodic HMM or fully

connected HMM, in which every state of the model can be reached from every other state in a single step [52]. The other type of HMM is the left-right HMM which has the property that as the time progresses, the state number increases, i.e., state transition proceeds from left to right and there is no return transition. This type of modeling puts a constraint on the state probability matrix such that:

$$a_{ij} = 0, \quad j < i$$

The left-right HMM model is suitable for the type of applications where states change over time. This applies to the problem of a multi-stage attack that progresses over time to reach the goal of compromising a certain target. Even though the attacker sometimes may perform actions that are supposed to lead to lower security states from the current state for reasons depend on the type of attack, the left-right HMM does not allow transition to a lower state. Therefore, two of the proposed architectures are modeled using a modified HMM model where right-left transitions are allowed for a specific cases.

2.2.2 HMM Parameters and Training

One of the important parts of any HMM-based architecture is the parameterization of HMMs in terms of determining both A_k and B_k matrices in (2.1) to maximize probability $Pr(O|\lambda_k)$ for each multi-stage attack. There are several unsupervised training algorithms, such as Baum-Welch (BW) and Expectation Maximization (EM) [52]. In this thesis, we use BW training algorithm as it is the most widely used algorithm and it is a special case of the EM method applied to HMM training [52]. Furthermore, since we have a limited availability of public datasets with multiple multi-stage attacks, we train HMMs with a small number of training data points, although practically the larger the number of training data points the better the model is, and thus the better detection accuracy can be obtained [48].

The training algorithm starts with an initial Markov model created and parameterized randomly (i. e. $\lambda_{initial} = (A_{initial}, B_{initial}, \pi)$). Then, the algorithm is executed

multiple times on a training sequence of observations and in every iteration of the algorithm, it improves the probability that the model $\lambda = (A, B, \pi)$ will match observation sequence O. That is,

$$P(O/\lambda_t) > P(O/\lambda_{t-1})$$

This process continues until convergence occurs, i.e., no further improvement for the probability is observed.

2.2.3 State Estimation and Prediction

In order to solve the second fundamental problem related to HMM (i.e., finding the best state path for a specific IDS alert observation or a window-sized sequence of alerts), we use the Viterbi algorithm. Viterbi algorithm computes the most probable path of states based on the state probability for each stage of the multi-stage attack. We calculate the multi-stage attack risk probability P_{MAS} and the progress of the attack based on the obtained state probability and samples of a similar multi-stage attack used in a supervised training phase as will be shown later. In particular, given a model λ_i, the objective is to find P_{MAS} by computing the most probable state the system is in after the last sequence of observations.

To apply this solution, we first define the variable $\gamma_t(i)$ (i.e., the probability of being in state S_i at time t) formally as follows:

$$\gamma_t(i) = P(x_t = s_i|O, \lambda_k) \tag{2.2}$$

This probability can be expressed in terms of the forward-backward variables as follows [52]:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O|\lambda_k)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i)} \tag{2.3}$$

$\alpha_t(i)$ accounts for the partial observation sequence $\{O_1, O_2, \ldots, O_t\}$ while $\beta_t(i)$ accounts for the remainder of the observation sequence $\{O_{(t+1)}, O_{(t+2)}, \ldots, O_T\}$ at time t and given the state S_i.

Considering one fixed state sequence $X = x_1, x_2, \ldots, x_T$. The Fordward-Backward consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(\{O_1, O_2, \ldots, O_t\}, x_t = s_i|\lambda_k) \tag{2.4}$$

In a similar manner, we can consider a backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(\{O_{t+1}, O_{t+2}, \ldots, O_T\}, x_t = s_i|\lambda_k) \tag{2.5}$$

The computation of the forward and backward variables is shown in [52] and it requires the order of $O(T \cdot N^2)$ calculations [49].

The normalization factor $P(O|\lambda_k) = \sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i)$ makes $\gamma_t(i)$ a probability measure and thus:

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{2.6}$$

We can solve then compute the most probable state x_t for each observation at time t as follows [52]:

$$x_t = \max_{1 \leq i \leq N} [\gamma_t(i)], 1 \leq t \leq T \tag{2.7}$$

Although (2.12) maximizes the expected number of correct states, if the HMM model has state transitions that have zero probability, this states sequence could not be correct because (2.12) determines the most likely state at any time, without taking into account the probability that the state sequence will occur. Therefore, we use the Viterbi algorithm to find the single best state sequence [52], $X = x_1, x_2, ..., x_T$, for the given observation sequence $O = o_1, o_2, ..., o_T$ from an IDS. The best score along a single path at time t is defined by $\delta_t(i)$, which considers the first t observations and ends in state s_i:

$$\delta_t(i) = \max_{x_1, .., x_{t-1}} P(x_1 \ldots x_t = s_i, o_1 \ldots o_t|\lambda_i) \tag{2.8}$$

The essential part of the Viterbi algorithm is that if we find $\delta_t(i)$, then by induction:

$$\delta_{t+1}(i) = [\max_i \delta_t(i) \cdot a_{ij}] \cdot q_j(o_{t+1}) \tag{2.9}$$

Finally, the Viterbi algorithm calculates the best state sequence. It has a complexity of $O(T \cdot N^2)$.

3. DESIGN AND PERFORMANCE EVALUATION OF A RESETTING LEFT-RIGHT HIDDEN MARKOV MODEL ARCHITECTURE (R-LRHMM)

The main objectives of this chapter are to design a detection architecture that estimates the current state of multi-stage attacks and to provide performance evaluation metrics for HMM-Based detection architectures.

3.1 Introduction

An HMM-based generic architecture for detecting multiple multi-stage attacks is shown in Fig. 3.1. The generic architecture employs information provided by alerts from intrusion detection systems (IDSs) that monitor the cyber-based infrastructure for malicious activities, conjointly with knowledge about the network topology and the functional dependencies between network assets. Based on real-time alert information, the HMM detection architecture is utilized to estimate the state and the progression of the attack and to predict the type and the real-time risk of an impending threat. The main challenges addressed in this chapter are as follows:

- Challenge 1: how to model each multi-stage attack in terms of HMM states and observations.
- Challenge 2: how to accurately estimate the security state of CBS in real-time during an impending threat and how to design an efficient architecture that detects and tracks the progress of multi-stage attacks.
- Challenge 3: how to accurately quantify and measure the detection performance of such an architecture, to predict the progression of the multi-stage attack, to

predict how far the attack is from its next stage or from its goal(s) and to estimate how fast it is.

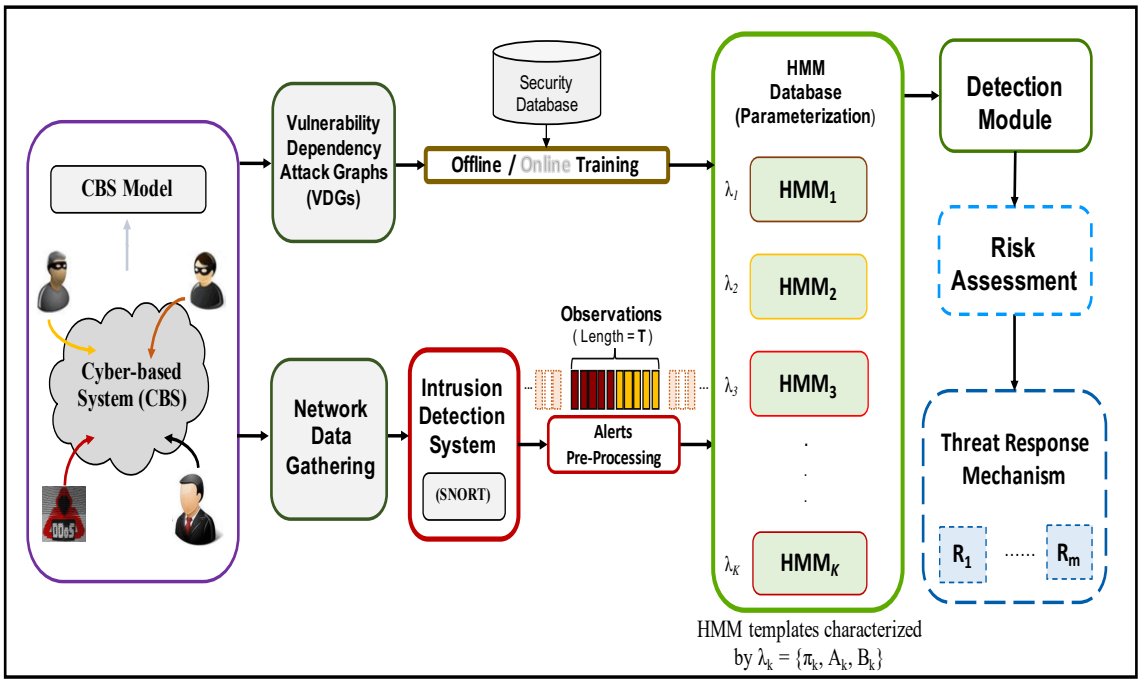


Fig. 3.1.: A Generic Architecture for Multiple Multi-stage Attack Detection using an HMM database

3.1.1 Objectives and Contributions

The main objectives and contributions of this chapter are as follows. First, we present the Resetting Left-Right HMM (R-LRHMM) architecture for detecting a single multi-stage attack or multiple multi-stage attacks occurring sequentially in a network using multiple HMMs. The design of R-LRHMM relies on modifying HMM model parameters to detect multiple multi-stage attacks. Further, for evaluation purposes, we propose three performance metrics, in addition to the widely used state probability metric [45]. The proposed metrics are: (1) the attack risk probability which provides insight to the speed of the attacks and can help in prioritizing re-

sponse actions, (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 performance metrics is given in the following subsections. The performance of the proposed architecture is evaluated using extensive simulation experiments to demonstrate its efficacy. These experiments are conducted using DARPA2000 public dataset as will be discussed later [8]. In the next section, the proposed architecture is described in more details.

3.2 The Resetting Left-Right Hidden Markov Model Architecture

The proposed R-LRHMM is shown in Fig. 3.2. 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. 3.2. Arrival of these alerts represents the interleaved attacks mentioned in Section III-B. The HMM_k 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 that among T observations, there are L_k observations (i.e., $\{o_{1_k}, o_{2_k}, \ldots, o_{L_k}\}$) belong to Attack k, and the remaining $T - L_k$ observations are considered by HMM_k as unrelated (interfering) alerts. We introduce a common state that encompasses all the unrelated alerts in HMM_k.

For HMM structure, we focus on the alerts generated by the IDS. In R-LRHMM, 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 (i.e., reset mode) 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 HMM parameters, (i.e., matrices A_k and B_k). This can be obtained by considering an observation o_t, such that

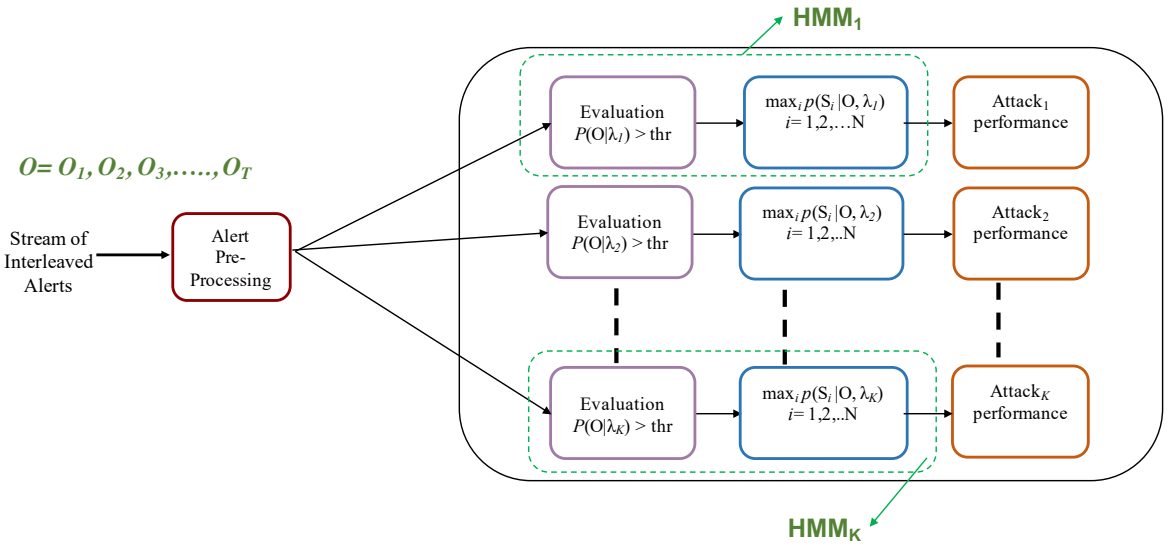


Fig. 3.2.: R-LRHMM Architecture

$\{o_t \notin V_k\}$. Accordingly, we add an extra column in the emission probability matrix, B_k, to account for this new symbol, as follows:

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

Note that transition to State 1, in the presence of unrelated observation o_t, occurs with probability ϵ_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 can be done by introducing transition probability (ϵ_2) in the first column of the A_k matrix.

Although our initial assumption is 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 it simplifies the training of each HMM. Subsequently, by introducing ϵ_2, the matrix A_k becomes as follows:

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

Based on this modification and training of the HMM template (λ_k), the evaluation module determines whether Attack k is active or not, as shown in Fig. 3.2, according to the criteria $Pr(O|\lambda_k) \geq thr$. Note that 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 in the range of 0 to 0.5. However, with the larger the value of thr, HMM template (λ_k) estimates only the states of the high probability sequences. In this chapter, we take a conservative approach in choosing $thr = 0$. The evaluation probability can be computed using the forward algorithm [52]. In case Attack k is active, then HMM$_k$ (λ_k) runs the Viterbi algorithm to decode the most probable hidden states that correspond to the given observation sequence $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$, as follows:

$$\begin{aligned} x_t &= \max_{1 \leq i \leq N_k} \gamma_t(i) \\ \gamma_t(i) &= Pr(x_t = s_i | O, \lambda_k) \\ t &= 1, \ldots, T \end{aligned} \tag{3.1}$$

where $\gamma_t(i)$ represents the probability of being in state s_i at time t based on the observation sequence. In R-LRHMM, each HMM template in Fig. 3.2 uses the Viterbi algorithm to find the best state sequence, $X = \{x_1, \ldots, x_t, \ldots, x_T\}$. For a

given observation sequence, Viterbi algorithm finds the highest probability along a single path for every o_t $(t \leq T)$ and o_t its corresponding state s_i such that:

$$\delta_t(i) = \underset{s_1,\ldots,s_{t-1}}{\text{argmax}} Pr(s_1, \ldots, s_t, o_1, \ldots, o_t | \lambda_k) \tag{3.2}$$

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

$$\delta_{t+1}(j) = \underset{1 \leq i \leq N_k}{\text{argmax}} \{\delta_t(i) a_{ij}(k)\}.b_i(o_{t+1}(k)) \tag{3.3}$$

This computation for a given sequence is repeated by all HMM templates in R-LRHMM (Fig. 3.2). Table 3.1 shows the overall processing of alerts based on R-LRHMM.

Table 3.1.: Detection process for R-LRHMM

Input: interleaved alerts: $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$,
π_k: $\lambda_k, k = 1, 2, \ldots, K$,
Output: $X = \{x_1, x_2, \ldots, x_T\}$

```
1   While (O is not empty)
2     for k = 1 : K
3       if (Pr(O|λ_k ≥ thr))
4         for t = 1 : T
5           Compute γ_t(i),   i = 1, 2, ..., N_k from equation (5.1)
6           x_t = max_{1≤i≤N_k} γ_t(i)
7         endfor
8       endif
9     endfor
10  endWhile
```

3.3 Performance Evaluation

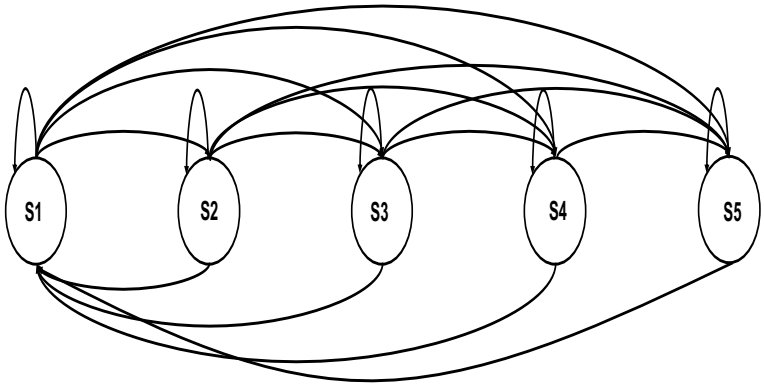
In this section, the performance evaluation of HMM-based alert prediction mechanism is conducted. Experiments are mainly focused on identifying the effect of the window size and HMM parameters on the accuracy of alert prediction.

3.3.1 Experimental Setup

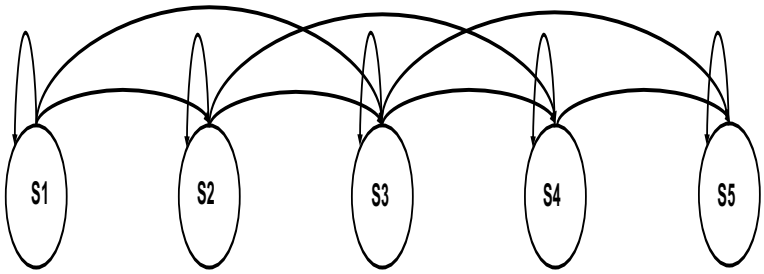
In this subsection, we discuss our experimental results based on the DARPA2000 dataset [8], which contains two DDoS multi-stage attacks labeled as LLDDOS 1.0 and LLDDOS 2.0.2. Both of these attacks have five stages, that can be summarized as IP sweeping, Sadmind probing, Sadmind exploit, DDoS software installation and Launching. Therefore, the training of the two HMMs is conducted based on a five state model ($N_k = 5, k = 1, 2$), which corresponds to five stages in each attack. The state diagrams of HMM1 for Attack 1 and HMM2 for Attack 2 are shown in Fig. 3.3. The parameterization of each HMM will be discussed later.

In our experiment, DARPA2000 raw network packets were processed by SNORT intrusion detection system [9] to generate alerts. The total number of alerts results 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, $M_k = 12, k = 1, 2$, as mentioned in Section II-A. The preprocessing module assigns a severity level to these alerts based on their relation to the stages of multi-stage attacks. In case there are more than one alert which lead to a state, the higher the severity level is given to the alert which indicates that the attack is progressing. Accordingly, Tables 3.2 and 3.3 show alert severity, alert type and states of both attacks [45].

One of the important parts of any HMM-based architecture is the parameterization of HMMs in terms of determining both A_k and B_k matrices to maximize probability $Pr(O|\lambda_k)$ for each multi-stage attack. We use BW training algorithm as it is the most widely used algorithm and it is a special case of the EM method applied to HMM training [52]. Furthermore, since we have a limited availability of



(a) HMM1



(b) HMM2

Fig. 3.3.: State Diagrams for HMM1 and HMM2

public datasets with multiple multi-stage attacks, we train HMMs with a small number of training data points, although practically the larger the number of training data points the better the model is, and thus the better detection accuracy can be obtained [48].

The training algorithm starts with an initial Markov model created and parameterized randomly (i.e. $\lambda_{initial} = (A_{initial}, B_{initial}, \pi)$). Then, the algorithm is executed multiple times on a training sequence of observations and in every iteration of the algorithm, it improves the probability that the model $\lambda = (A, B, \pi)$ will match observation sequence O. That is,

$$P(O/\lambda_t) > P(O/\lambda_{t-1})$$

Table 3.2.: Correspondence between Alert type, Alert Severity and States in DARPA LLDDOS 1.0

Alert Severity	Alert Type	State
1	ICMP PING	1
2	ICMP Echo Reply	1
3	ICMP PING BSDtype	1
4	ICMP PING Unix	1
5	RPC portmap sadmind request UDP attempt	2
6	ICMP Destination Unreachable Port Unreachable	2
7	RPC sadmind UDP PING	2
8	RPC sadmind query with root credentials attempt UDP	3
9	RPC sadmind UDP NETMGT CLIENT overflow attempt	3
10	SERVICES rsh root	4
11	TELNET login	4
12	SNMP AgentX/tcp request and flood DDoS attempt	5

This process continues until convergence occurs, i.e., no further improvement for the probability is observed.

For simplicity, we assume five stages in our training model for both attacks. Alerts are mapped into five different sets which are the same as states in our HMM. For example, an alert of ICMP PING type is usually considered as a scanning/probing type and it is converted to its corresponding stage which is scanning/probing. This mapping is used to train each HMM state for a group of alerts (observations). After forming five sets of alerts that are mapped to five states of an HMM model, we use BW algorithm to generate the HMM parameters $\lambda = (A, B, \pi)$.

The two HMMs are trained using 30% of the DARPA2000 dataset, i.e. HMM1 is using LLDDOS 1.0 and HMM2 is using LLDDOS 2.0.2 for training, while the rest of the dataset was used for testing.

Table 3.3.: Correspondence between Alert type, Alert Severity and States in DARPA LLDDOS 2.0.2

Alert Severity	Alert Type	State
1	RPC portmap sadmind request UDP	1
2	RPC portmap Solaris sadmind port query udp request	1
3	RPC portmap Solaries sadmind port query	1
4	udp portmapper sadmind port query attempt	1
5	RPC sadmind query with root credentials attempt UDP	2
6	RPC sadmind UDP	2
7	sadmind UDP NETMGT CLIENT overflow attempt	2
8	RPC portmap Solaries sadmind port query	3
9	RPC sadmind query with root credentials attempt UDP	3
10	RPC sadmind UDP - Diff. IP	4
11	sadmind UDP NETMGT CLIENT overflow attempt	4
12	ICMP Destination Unreachable Port Unreachable and flood DDoS attempt	5

The training of HMMs in terms of computation of observations and states probabilities are based on the machine learning toolbox in MATLAB [53]. For the purpose of training, we use 30% of the dataset, while the rest of the dataset was used for testing. In this evaluation, parameters ϵ_1 and ϵ_2 of models λ_1 and λ_2 of R-LRHMM are chosen as $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-3}$.

The main parameters of HMM model in both attacks, which are determined from training using BW algorithm are as follows:

$$A_1 = \begin{bmatrix} 0.8550 & 0.1448 & 0.0001 & 0.0001 & 0.0001 \\ 0 & 0.9 & 0.0997 & 0.0001 & 0.0001 \\ 0 & 0 & 0.9880 & 0.0090 & 0.0001 \\ 0 & 0 & 0 & 0.9880 & 0.0090 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0.9287 & 0.0712 & 0.0001 & 0 & 0 \\ 0 & 0.9141 & 0.0857 & 0.0001 & 0 \\ 0 & 0 & 0.9387 & 0.0612 & 0.0001 \\ 0 & 0 & 0 & 0.9752 & 0.0248 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0.1131 & 0.1342 & 0.1397 & 0.1610 & 0.0861 & 0.1093 & 0.0955 & 0.0407 & 0.0999 & 0.0656 & 0.0970 & 0.0467 \\ 0.0884 & 0.1096 & 0.1335 & 0.0355 & 0.1185 & 0.1230 & 0.1266 & 0.1088 & 0.0803 & 0.0965 & 0.0593 & 0.0333 \\ 0.0095 & 0.0638 & 0.1138 & 0.1163 & 0.0201 & 0.1162 & 0.0536 & 0.1211 & 0.1127 & 0.1165 & 0.0577 & 0.0451 \\ 0.0719 & 0.1338 & 0.0688 & 0.1016 & 0.0796 & 0.0504 & 0.0487 & 0.0690 & 0.1016 & 0.1275 & 0.1382 & 0.0664 \\ 0.0297 & 0.0707 & 0.0771 & 0.0947 & 0.0349 & 0.1927 & 0.0115 & 0.0844 & 0.0553 & 0.0032 & 0.0207 & 0.1142 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0.1376 & 0.0291 & 0.0033 & 0.1538 & 0.1174 & 0.0049 & 0.0044 & 0.1838 & 0.1490 & 0.0304 & 0.1501 & 0.0361 \\ 0.0922 & 0.0651 & 0.1968 & 0.0427 & 0.1096 & 0.0941 & 0.1127 & 0.0089 & 0.0886 & 0.0358 & 0.0800 & 0.0736 \\ 0.0630 & 0.0568 & 0.1421 & 0.0052 & 0.0007 & 0.1598 & 0.1049 & 0.0798 & 0.0519 & 0.1332 & 0.0955 & 0.1071 \\ 0.0459 & 0.1605 & 0.1204 & 0.1586 & 0.0158 & 0.0075 & 0.1319 & 0.1817 & 0.1053 & 0.0160 & 0.0521 & 0.0041 \\ 0.0356 & 0.0408 & 0.0978 & 0.0841 & 0.0290 & 0.1304 & 0.0411 & 0.1242 & 0.1427 & 0.0693 & 0.0357 & 0.1693 \end{bmatrix}$$

The training of HMMs in terms of computation of observations and states probabilities are based on the machine learning toolbox in MATLAB [53]. For the purpose of training, we use 30% of the dataset, while the rest of the dataset was used for testing. In this evaluation, parameters ϵ_1 and ϵ_2 of models λ_1 and λ_2 of R-LRHMM are chosen as $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-3}$.

For the completeness of our evaluation, we present the detection performance of the two multi-stage attacks using their respective HMMs when these attacks occur one at a time. Subsequently, interleaved scenarios of the two attacks can be generated with varying the starting point of the interleaving to test the performance of the proposed architecture. For all the results, the x-axis shows the running count of alerts as they are generated by SNORT. For the purpose of evaluation, we propose

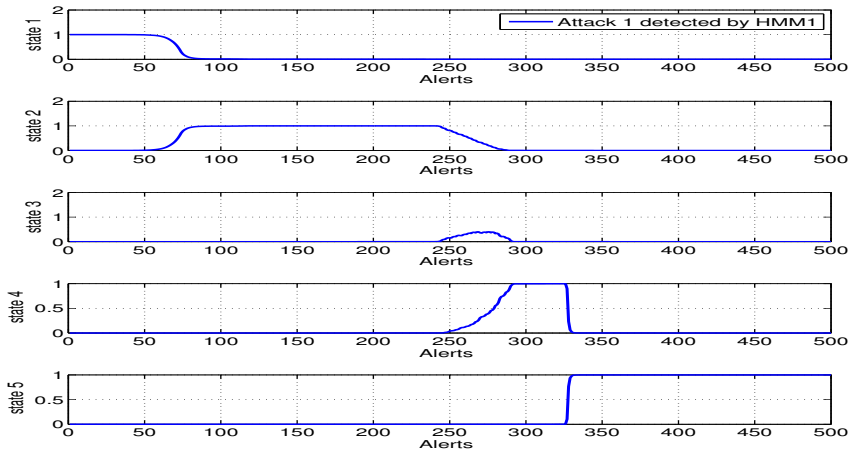
three performance metrics, in addition to the widely used state probability metric [45]. The performance metrics are: the attack risk probability which provides an insight about the speed of attacks and can help in prioritizing response actions; and the number of correctly detected stages. The justification of these performance metrics is given in the following subsections.

3.3.2 Detection of Individual DARPA2000 Attacks

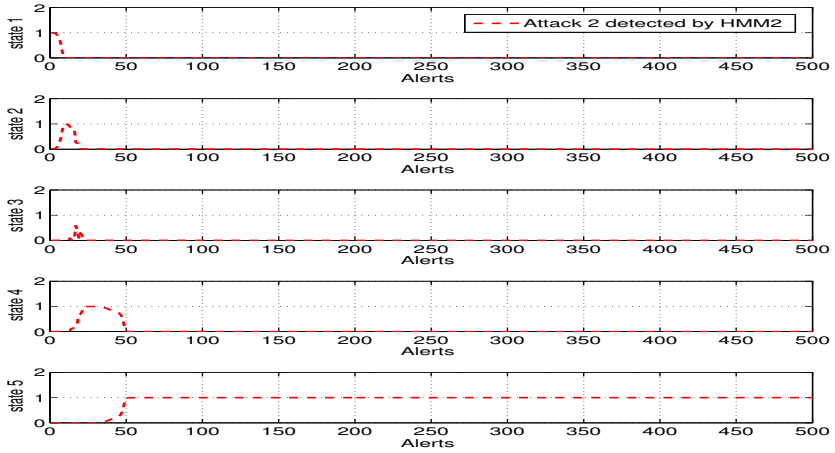
In this subsection, we first consider the alerts generated by SNORT when only LLDDOS 1.0 dataset is used. We pass these alerts to the alerts preprocessing module and then to HMM1 which is trained to detect the first DDoS attack (Attack 1). The same is done for LLDDOS 2.0.2 dataset to detect the second DDoS attack (Attack 2). The state probability results of each attack is shown in Fig. 3.4. It can be noted that the five stages of each attack are detected by each HMM. It can also be noted that Attack 2 is relatively faster than Attack 1, as few alerts are needed by Attack 2 to reach the last stage of HMM2 indicating the launching of the DDoS attack on the target. In Fig. 3.4 we show the estimated state corresponding to only the first 500 alerts out of all the SNORT alerts as the remaining alerts are almost all the same and lead to the compromise state (State 5). The observation length for this case is taken as $T = 100$.

3.3.3 Generating Interleaved Scenarios

Based on the two multi-stage attacks in the DARPA2000 dataset, we altered the timestamp of some of 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 a network. Fig. 3.5 shows a scenario of interleaved alerts for two simultaneous DDoS attacks. Since Attack 2 takes shorter



(a) Attack1



(b) Attack2

Fig. 3.4.: State Probability of HMM1 and HMM2 for Individual DARPA Attacks

time to compromise the target and launch DDoS, we manipulate timestamps of Attack 2 so that it starts in the middle of Attack 1. The y-axis in Fig. 3.5 represents the alert severity based on the preprocessing module as depicted in Tables 3.2 and 3.3. Based on this scenario, the performance results of the proposed architecture are given in the following subsections.

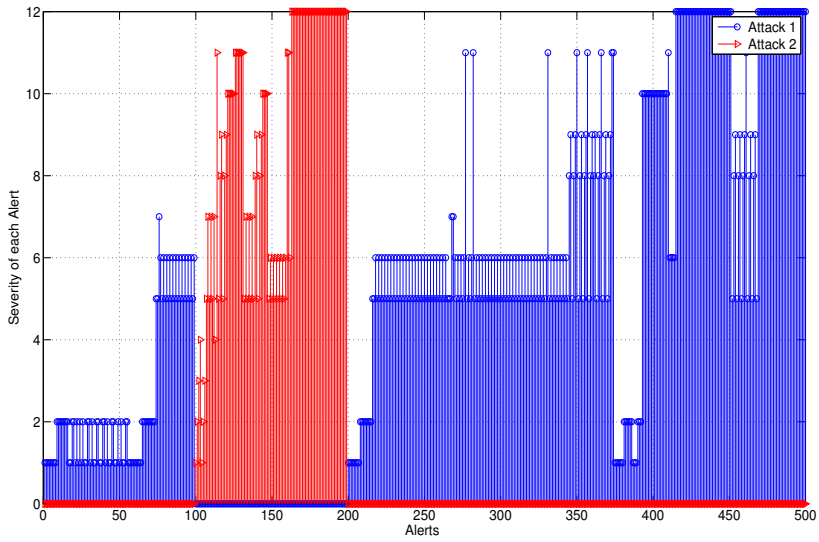


Fig. 3.5.: Scenario of Interleaved Alerts from LLDDOS 1.0 and LLDDOS 2.0.2 Attacks

3.3.4 Probability of State Estimation and the Effect of Interleaving

In this subsection, we present the state probability, $\gamma_t(i)$, from (5.1) and (4.2) for $i = 1, \ldots, 5$ with two observation lengths, $T = 10$ and $T = 500$. For $T = 500$, it can be seen from Fig. 3.6 that R-LRHMM can estimate[1] the states of both attacks with a high probability, especially for States 1, 2, 4, and 5. Note that the state probability for State 3 of Attack 1 is very low for both values of T. The reason is there are not enough alerts produced by SNORT for this stage.

We observe discontinuity in the state probability plot of R-LRHMM in Figs. 3.6 and 3.7, which is due to the fact that both HMMs return to State 1 whenever there exist interfering alerts from other attacks. Fig. 3.8 shows the importance of considering ϵ_2 in designing HMM used by R-LRHMM. 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 that $\epsilon_2 = 0$ represents the case of generic architecture, for which returning to State 1 is not allowed when

[1]Note: The terms detecting a state and estimating a state are used interchangeably in this chapter

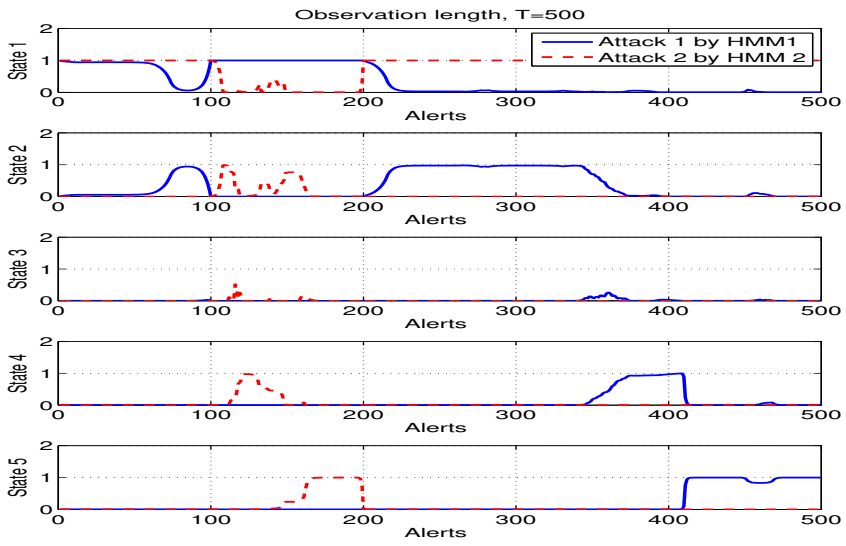


Fig. 3.6.: State probability of Attacks 1 and 2 detected by HMM1 and HMM2 based on R-LRHMM, T=500

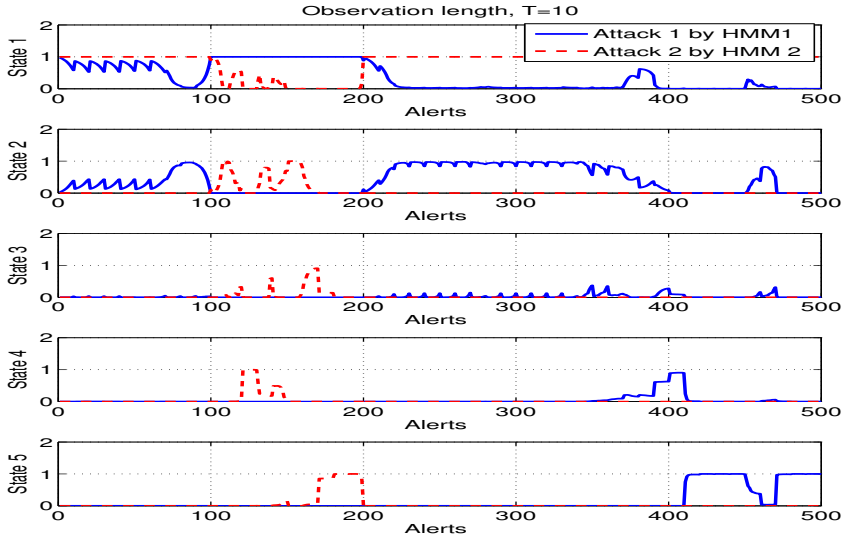


Fig. 3.7.: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 based on R-LRHMM, T=10

HMM1 receives alerts belong to Attack 2 or when HMM2 receives alerts belong to Attack 1. Setting $\epsilon_2 = 0$ reduces the accuracy of state detection for the two attacks, as can be seen in Fig. 3.8. For example, Fig. 3.8 provides no estimate for state probability of States 2 and 4 for the first 200 alerts when $\epsilon_2 = 0$, as compared to Fig. 3.8 when $\epsilon_2 = 0.001$. Similar observation can be made by comparing Figs. 3.8 and 3.8 for the first 350 alerts of State 2. In summary, Figs. 3.6 and 3.7 show that there is no significant change in the state detection performance of R-LRHMM as the observation length changes from 10 to 500 alerts.

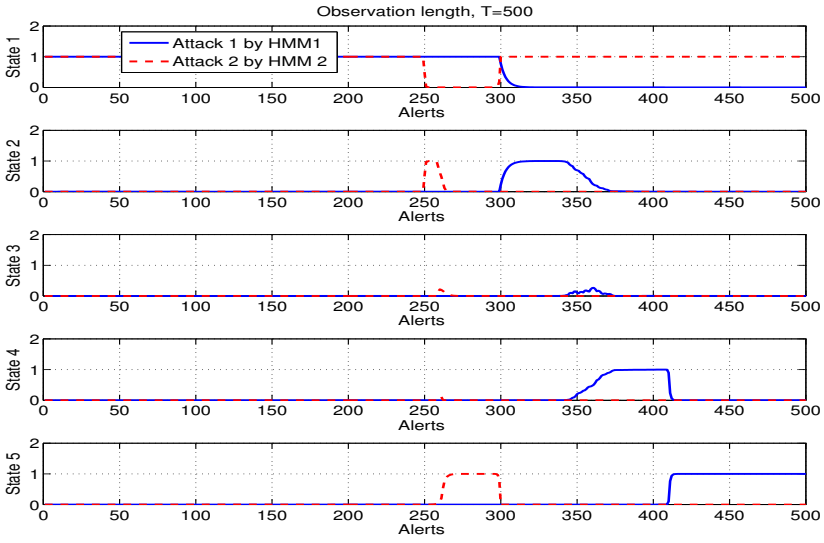


Fig. 3.8.: Effect of ϵ_2 on State Probability based on R-LRHMM

3.3.5 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 ($\gamma_t(i)$) averaged over the total number of states. Its value

gets updated at every observation length in a non-decreasing manner, as shown below in (3.4):

$$Pr_{attack_k}(t) = \frac{\sum_{i=1}^{N_k} \gamma_t(i) s_i}{N_k} \qquad t = 1, \ldots, T \;\; i = 1, \ldots, N_k, \;\; k = 1, \ldots, 2 \tag{3.4}$$

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. 3.9 and 3.10 show the attack risk probability for both DARPA multi-stage attacks using R-LRHMM for the two observation lengths, $T = 10$ and $T = 500$. Note that there is no significant difference between the case of $T = 10$, and $T = 500$. Also Note that after 100 alerts, Attack 2 progresses relatively fast, and reaches the compromise state well before Attack 1.

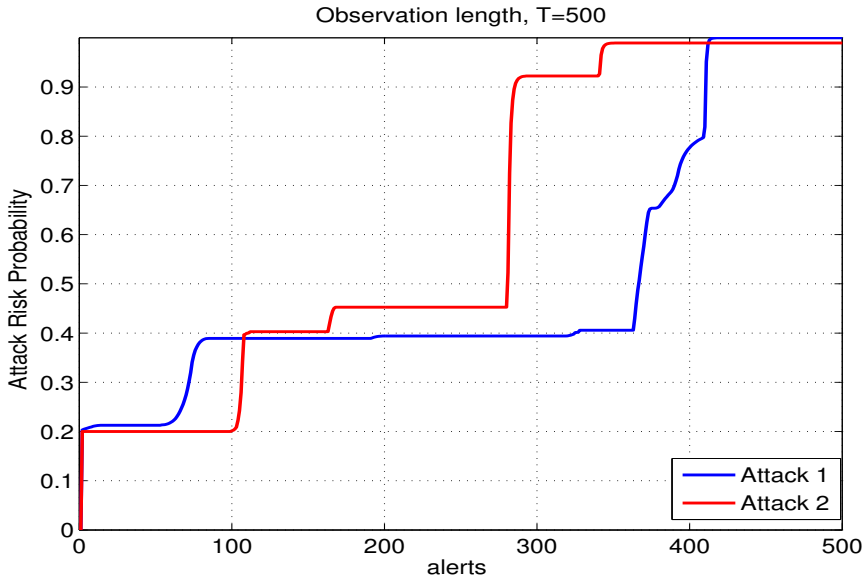


Fig. 3.9.: Attack risk probability based on R-LRHMM where T = 500

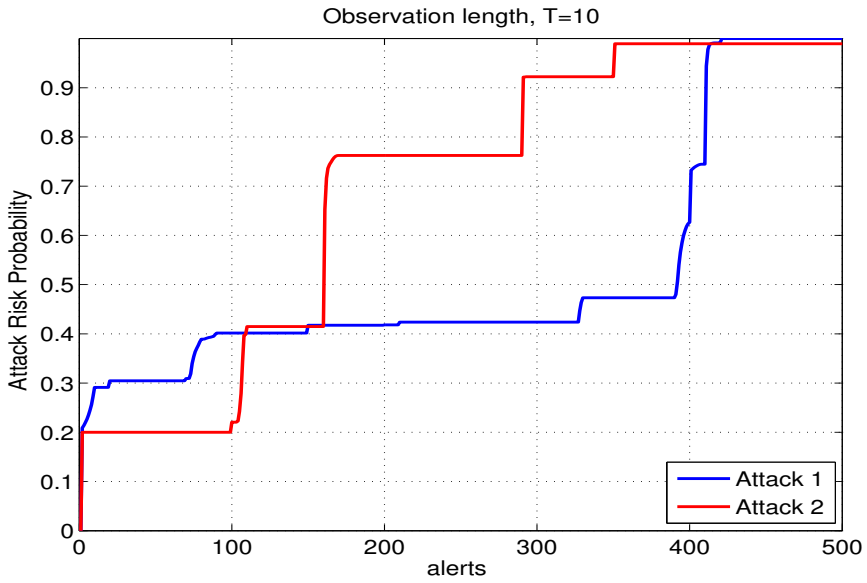


Fig. 3.10.: Attack risk probability based on R-LRHMM where T = 10

3.3.6 Number of Correctly Detected Stages of Multi-stage Attacks

The next 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 from the point of view of considering response actions. We evaluate R-LRHMM by computing the number of detected stages per attack as follows. As we know the correspondence between alerts and stages (or states) of the multi-stage attack based on the knowledge of the DARPA2000 dataset, we compare the estimated states from each HMM with the exact states. Table 3.4 provides the results for three different values of T.

Table 3.4.: Number of correctly detected stages

Interleaving Scenario	**Architecture**	**Attack**	$T = 10$	$T = 100$	$T = 500$
Scenario	I	Attack 1	4	4	4
		Attack 2	5	5	5

3.3.7 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:

$$ER = \frac{\text{Number of incorrect detected states of the incoming Alerts}}{\text{Total number of Alerts}} \times 100 \qquad (3.5)$$

3.4 Conclusion

To sum up, the main objectives and contributions of this chapter are as follows. First, we propose a framework for detecting/predicting multiple multi-stage attacks occurring simultaneously in a network using HMM (sequentially or interleaved). Second, for the performance assessment of R-LRHMM, we propose three performance metrics. The performance metrics are: (1) the attack risk probability which provides insight to the speed of the attacks and can help in prioritizing response actions, (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 DARPA2000 dataset is chosen to demonstrate the efficacy of the proposed architecture.

The limitation with R-LRHMM is that there is a high probability of high false negatives in states detection, especially when the attacks are highly interleaved. The reason is that, each HMM template processes an observation sequence that contains interfering observations belonging to other attacks. However, the low performance of R-LRHMM is observed only in special attack scenarios. Nevertheless, R-LRHMM has a low computation complexity in terms of observations preprocessing as will be discussed later.

To address the challenge of the high degree of interleaving and to achieve a better performance, we propose another variation of the generic architecture. The new archi-

tecture is discussed in the next chapter. The aforementioned performance metrics will be used to compare between the performance of the proposed the two HMM-based architectures.

4. LEFT-RIGHT HIDDEN MARKOV MODEL ARCHITECTURE WITH ATTACK DEMULTIPLEXER (LRHMM+AD)

As mentioned in the previous chapter, the limitation with R-LRHMM is that there is a high probability of high false negatives in states detection when the attacks are highly interleaved. To address the challenge of the high degree of interleaving and to achieve a better performance, we propose in this chapter another variation of the generic architecture. The design of the proposed architecture, termed as LRHMM+AD, relies on de-interleaving mixed alerts from different attacks prior to the HMM processing subsystem. Furthermore, we compare the Left-Right Hidden Markov Model Architecture with Attack Demultiplexer (LRHMM+AD) with R-LRHMM in terms of their detection performance and design complexity.

4.1 Introduction

4.1.1 Using HMM to Detect Interleaved 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 [8].

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.

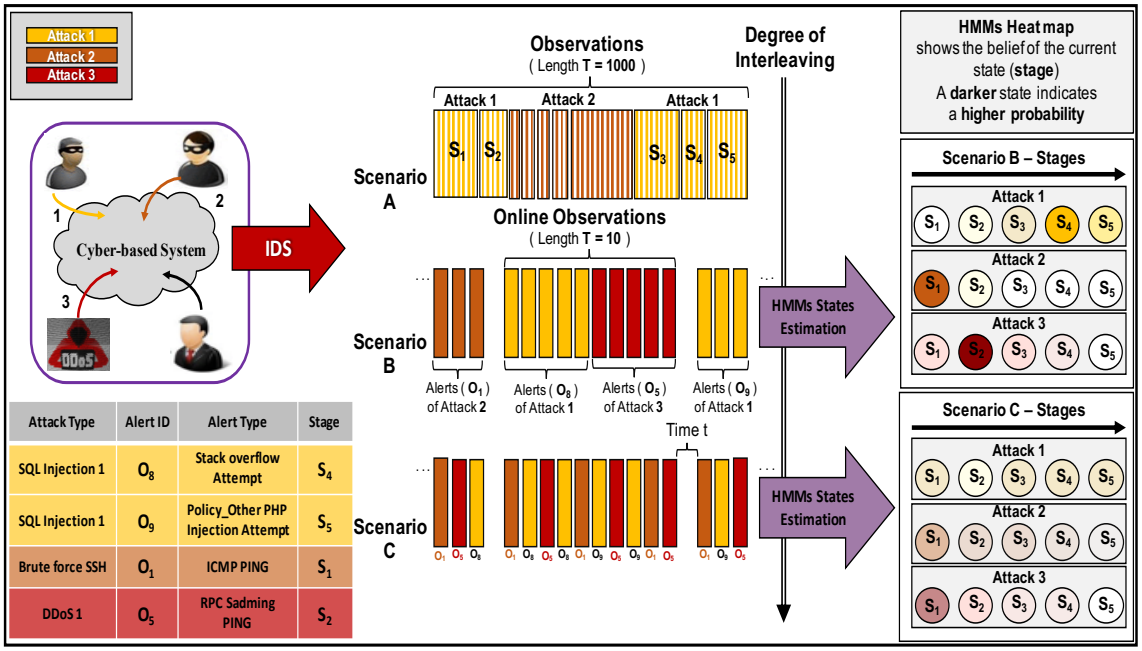


Fig. 4.1.: State Estimation of Multi-stage Attacks with Different Degree of Interleaving at Time t

Fig. 4.1 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 [8]. 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. 4.1 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. 4.1 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 varying the degree of interleaving is given later.

In this chapter, 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 that 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 λ_k. In the case of M possible attacks, we have a set of M templates.

Note that 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 [52]. In this chapter, 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 that 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 IV.

The parameters of the HMM template (i.e. the HMM model λ_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 [54]. 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 [52]. 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.

4.2 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. 4.2, 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 that 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 [9,55]), 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 [56]. The selection of IDS rules can help to reduce the large volume of alerts and false positives by tuning these rules. [10]. 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 that indicates indicates that an ongoing multi-stage attack is progressing towards an advanced stage. In this chapter, 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 that 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 [5].

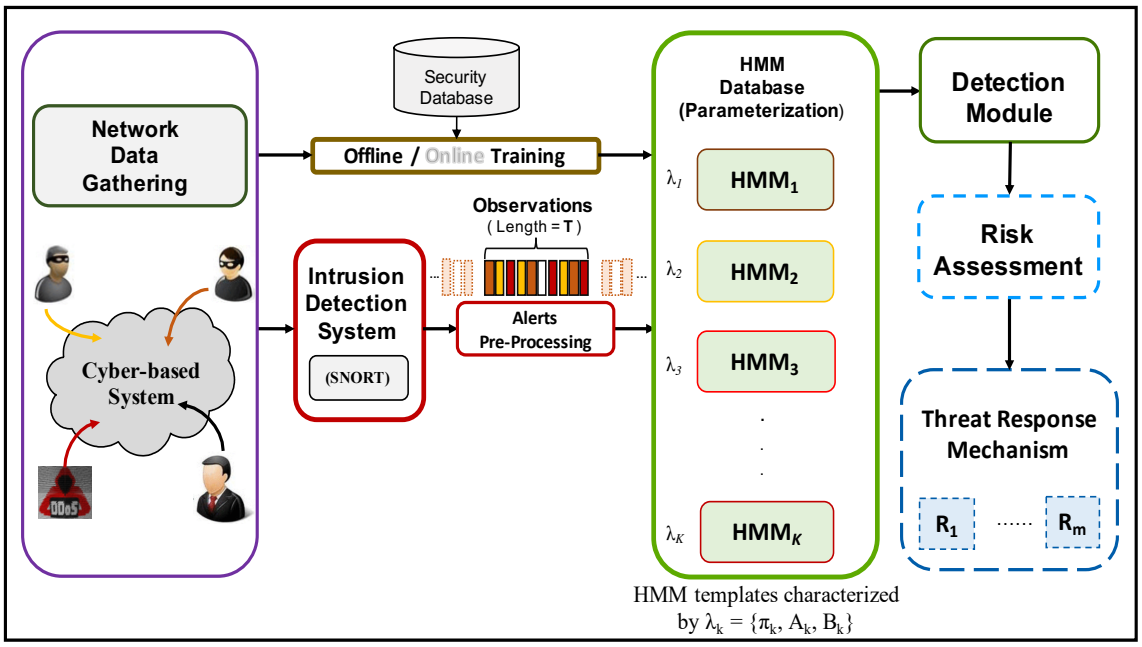


Fig. 4.2.: A Generic Architecture for Multiple Multi-stage Attack Detection Using an HMM database

4.2.1 Modeling Interleaved Attacks

Note that in general, K distinct multi-stage attacks can be launched simultaneously in a network, and their related alerts, generated by 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 that 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. 4.1.
- Multiple attacks start and finish at different times in the presence of one or multiple ongoing attacks.
- Stages of one attack can be embedded at different times across another ongoing attack(s).
- Stages of multiple attacks 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. 4.1).

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 [8]. 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. 4.2. The design of the first architecture, R-LRHMM, is based on modifying the HMM model parameters so that they can deal with the interleaved alerts. The design of LRHMM+AD improves attack detection capability by separating alerts from the various attacks prior to routing the alerts to HMM templates sub-system.

4.3 The Left-Right HMM Architecture with Attack Demultiplexer

As mentioned in the previous chapter, R-LRHMM 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 V. 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 R-LRHMM is observed only in special attack scenarios. Nevertheless, R-LRHMM has a low computation complexity in terms of observations preprocessing (as discussed later).

To achieve better performance, we propose another variation of the generic architecture of Fig. 4.2. Termed as LRHMM+AD, this new architecture, is depicted in Fig. 4.3 and is discussed below.

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 LRHMM+AD, we use these features to improve detection efficiency of the HMM templates. In particular, unrelated observations that do not belong to the k^{th} attack are separated and passed to their respective HMMs. Note that the major design philosophy behind LRHMM+AD is to use aforementioned features to preprocess the online network traffic stream and demultiplex it into multiple substreams, as shown in Fig. 4.3. Note each substream is routed to individual instances (planes) where each instance plane contains templates of all attack types. We refer to this preprocessing step as a demultiplexing step. It is an important step as it helps in eliminating the number of interfering observations from other attacks that are not detectable by a particular HMM.

Note that alerts triggered by the same attack scenario are correlated based on some features, (e.g., the source and destination IP addresses). We define alert (o_i) as a 7-tuple features (timestamp, ID, srcIP, srcPort, desIP, desPort, priority) according to the IDMEF [57], [58]. We refer to this tuple as a feature set $F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$. The timestamp represents date and time of an alert generated by the IDS. ID is the identification of the alert, srcIP and srcPort indicate

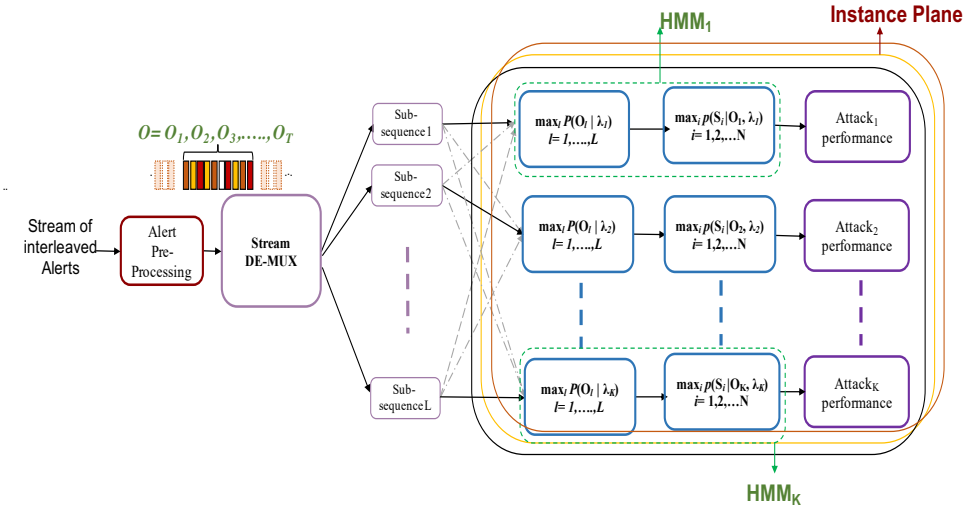


Fig. 4.3.: LRHMM+AD Exhibiting L Instance Planes with Substream Routing

the source IP address and source port number, respectively. Also, desIP and desPort represent the destination IP address and port number, respectively, and priority indicates the alert's rank [33]. Note that these features are used to distinguish between attacks as to 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 that the IP addresses of the alerts, which are triggered by the same attack scenario, are generally related in form 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 subsreams ($1 \leq L \leq K$). The demultiplexing process is based on one or more of the aforementioned distinguishing feature(s) of the incoming alerts and/or based on the correlation of IP addresses. Therefore, from the incoming stream of alerts, $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$, 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, $O_1 = \{o_{1_1}, o_{2_1}, \ldots, o_{T_1}\}$, $O_k = \{o_{1_k}, o_{2_k}, \ldots, o_{T_k}\}$, and so on till $O_L = \{o_{1_L}, o_{2_L}, \ldots, o_{T_L}\}$, where $L \leq K$ and $T_k \leq T$. Note that 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 that 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 LRHMM+AD, each HMM can have L instances, each of which can process one single substream. Thus, all the L generated substreams pass through each HMM to find which subsequence matches a certain HMM. The computation by each instance is performed based on the posterior probability given in (4.1).

$$O_k^* = \max_{1 \leq l \leq L} \quad Pr(O_l|\lambda_l), \quad L \leq K \tag{4.1}$$

Note that this probability computation is performed $L \times K$ times. The next stage of the HMM process is to estimate the state probabilities for its corresponding subsequence, O_k^*, found from (4.1) using the Viterbi decoding algorithm, as follows:

$$\begin{aligned} x_t &= \max_{1 \leq i \leq N_k} \gamma_t(i) \\ \gamma_t(i) &= Pr(x_t = s_i|O_k, \lambda_k) \\ t &= 1, \ldots, T_k \end{aligned} \tag{4.2}$$

Unlike R-LRHMM, the first stage of every HMM in LRHMM+AD has a maximum of K instances of the forward algorithm and one instance of the Viterbi algorithm.

Table 4.1.: Detection process for LRHMM+AD

Input: interleaved alerts: $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$,
π_k: $\lambda_k, k = 1, 2, \ldots, K$,
Output: $X = \{x_1, x_2, \ldots, x_T\}$

While (O is not empty)
  Demultiplex sequence O into L subsequences, $O_1, O_2, \ldots, O_L$,
    using features and address correlation
  for $k = 1 : K$
    for $l = 1 : L$
      Compute $(Pr(O_l|\lambda_k))$
    endfor(l)
    Find $O_k^* = \max_{1 \le l \le L} Pr(O_l|\lambda_k)$
    for $t = 1 : T_k$, T_k is the length of sequence O_k^*
      Compute $\gamma_t(i) = Pr(x_t = s_i|O_k^*, \lambda_k)$, from equation (4.2)
      $x_t = \max_{1 \le i \le N_k} \gamma_t(i)$
    endfor(t)
  endfor(k)
endWhile

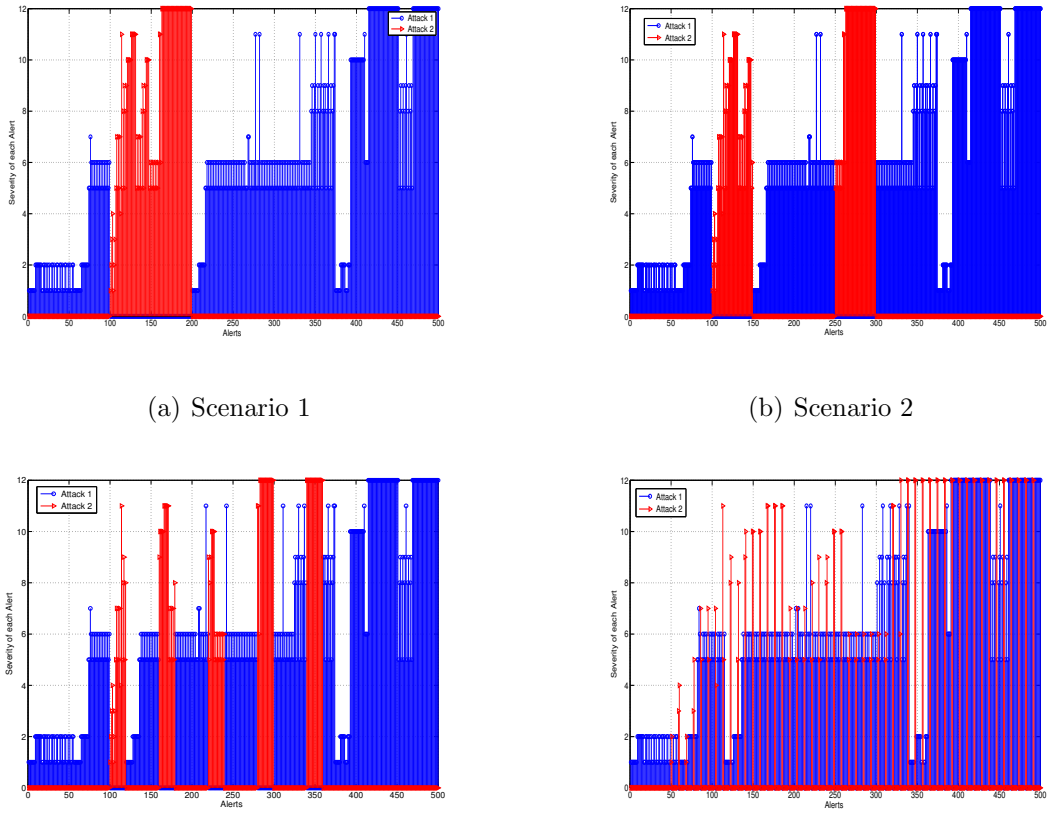
In addition, the HMM in LRHMM+AD deals with subsequences of length T_k, where $T_k \le T$. Table 4.1 shows the overall processing of alerts based on LRHMM+AD.

4.3.1 Complexity Comparison of the Proposed Architectures

Note that in R-LRHMM and LRHMM+AD in Figs. 3.2 and 4.3, the main modules that contribute to their computational complexity are the alert preprocessing

module for assigning alert severity, the stream demultiplexing module, and the HMM parallel branch modules. The first preprocessing module is the same for both architectures. However, the demultiplexing module exists only in LRHMM+AD, which demultiplexes all T alerts based on a subset (S) of alert features considered in the demultiplexing operation. The larger the T and S sets are, the more complex computation is performed by this module. In other words, as a result of the demultiplexing operation, LRHMM+AD has $T \times |S|$ additional computational steps as compared with R-LRHMM.

Next, we consider the HMM database component of the architectures. Note that 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 R-LRHMM, 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 LRHMM+AD, each HMM template processes, on the average, with a shorter sequence length compared to the sequence lengths in R-LRHMM. 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 LRHMM+AD, $L \times K$ computations of the FW algorithm plus L computations of the Viterbi algorithm are performed. It is important to note that although LRHMM+AD 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 R-LRHMM. The primary shortcoming of LRHMM+AD 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.



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 4.4.: Interleaved Alerts Scenarios from LLDDOS 1.0 and LLDDOS 2.0.2 Attacks

4.4 Evaluation and Performance Measures

In this section, we discuss the experimental results based on the DARPA2000 dataset [8], 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 [9] 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.

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. 4.2, 3.2, and 4.3. The reason for using the generic architecture of Fig. 4.2 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 utilize three performance metrics, in addition to the widely used state probability metric [45]. The performance metrics are: (1) the attack risk probability which provides insight to the speed of the attacks and can help in prioritizing response actions, (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 performance metrics is given in the following subsections.

4.4.1 Generating Alert Interleaving Scenarios

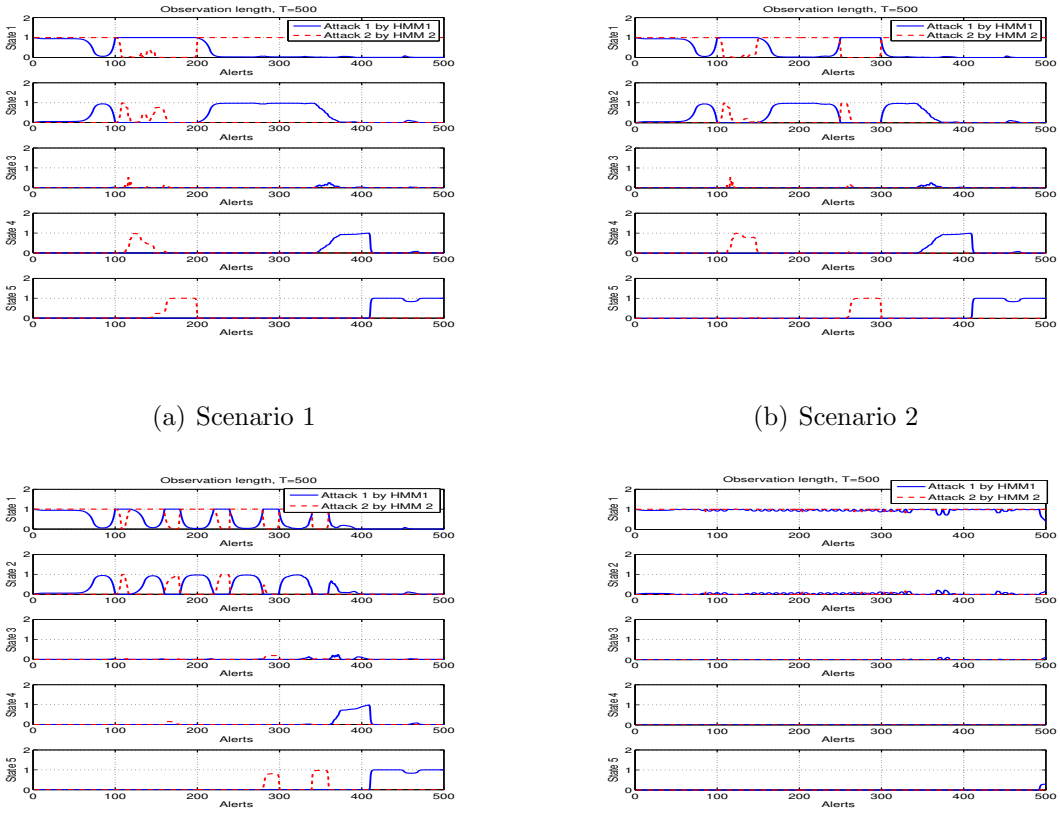
Based on the two multi-stage attacks in the DARPA2000 dataset, we alter the timestamp of some of these alerts in both attacks so that we can generate a single sequence of alerts that is composed of a mix of the two attacks without altering the temporal order of the original alerts. In addition, the IP 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. 4.4

shows several scenarios of interleaved alerts for two simultaneous DDoS attacks. Note that 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. 4.4 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.

4.4.2 Probability of State Estimation and the Effect of Interleaving

In this subsection, we present the state probability, $\gamma_t(i)$, from (5.1) and (4.2) for $i = 1, \ldots, 5$ with two observation lengths, $T = 10$ and $T = 500$. Regarding $T = 500$, it can be seen from Figs. 4.5(a) and 4.5(b) that R-LRHMM can estimate[1] 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, R-LRHMM fails to detect many states. For example, for Scenario 3, States 3 and 4 of Attack 2 are not detected, as depicted in Fig. 4.5(c). For Scenario 4, R-LRHMM performs very poorly as it fails to detect all the states of both attacks, as depicted in Fig. 4.5(d). For $T = 10$, R-LRHMM shows a small improvement in detecting States 3 and 4 for Scenarios 1 and 3, respectively, as can be seen from Fig. 4.5 and Fig. 4.7. The reason for the poor performance of R-LRHMM 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.

[1]Note: The terms detecting a state and estimating a state are used interchangeably in this chapter



(a) Scenario 1

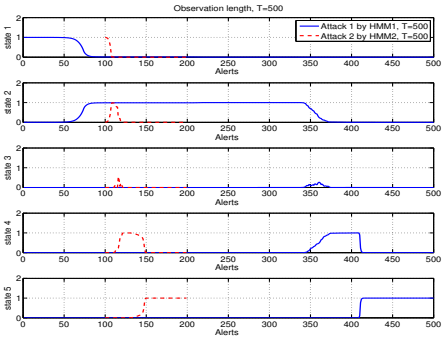
(b) Scenario 2

(c) Scenario 3

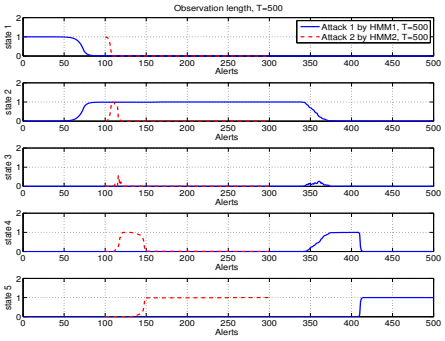
(d) Scenario 4

Fig. 4.5.: State probability of Attacks 1 and 2 detected by HMM1 and HMM2 based on R-LRHMM, T=500

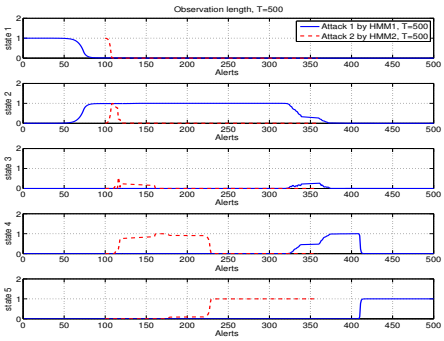
LRHMM+AD, on the other hand, performs better as compared to R-LRHMM in terms of estimating correct states of all incoming alerts for both $T = 10$ and $T = 500$. This performance improvement, even for higher degrees of interleaving, can be observed from Figs. 4.5(c), 4.5(d,) 4.6(c), 4.6(d), 4.7(c), 4.7(d), 4.8(c), and 4.8(d). The reason behind this performance improvement for LRHMM+AD is the presence of the demultiplexing module that helps each HMM to process only relevant attack alerts. Note that 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.



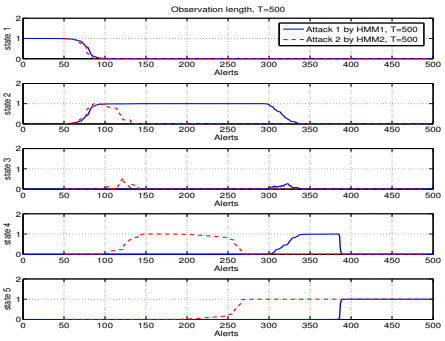
(a) Scenario 1



(b) Scenario 2



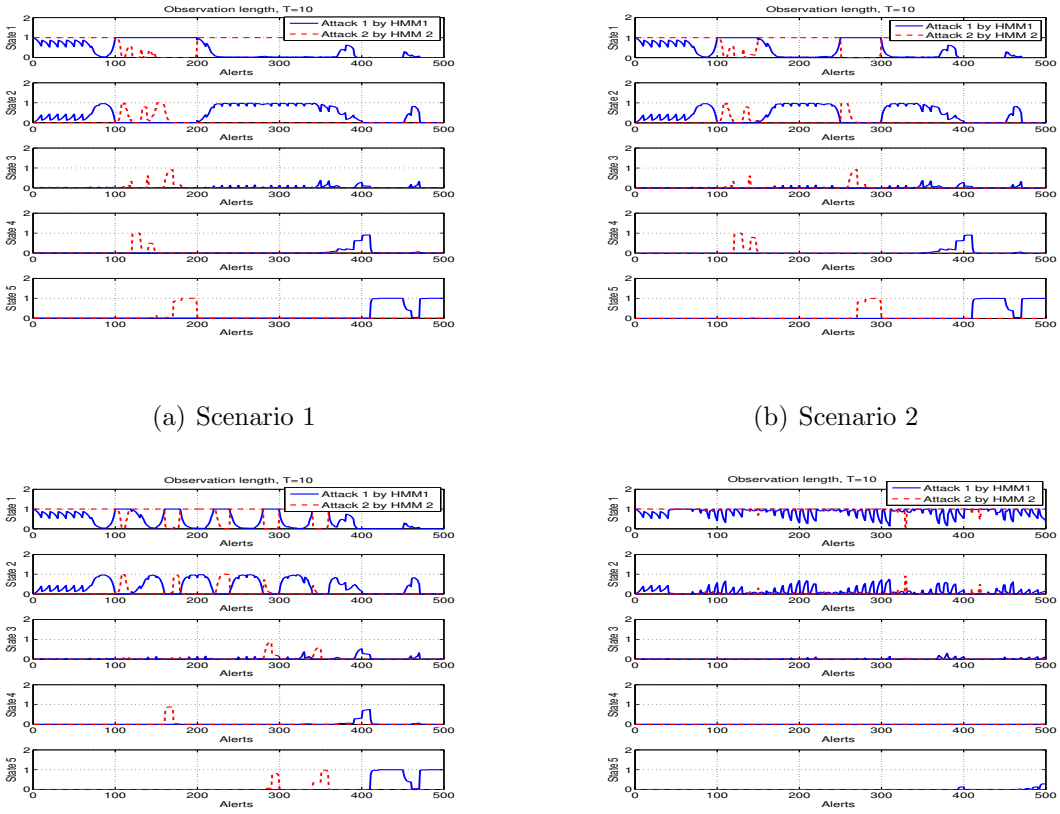
(c) Scenario 3



(d) Scenario 4

Fig. 4.6.: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 based on LRHMM+AD, T=500

We observe discontinuity in the state probability plot of R-LRHMM in Figs. 4.5 and 4.7, a condition that results when both of the HMMs return to State 1 whenever interfering alerts exist from other attacks. However, in LRHMM+AD, as the alerts from different attacks are demultiplexed prior to their processing by the HMMs, the states of the HMMs are not interrupted. Fig. 4.9 shows the importance of considering ϵ_2 in designing the HMM used by R-LRHMM. 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 that $\epsilon_2 = 0$ represents the case of generic architecture, for which returning to State 1 is not allowed when HMM1



(a) Scenario 1

(b) Scenario 2

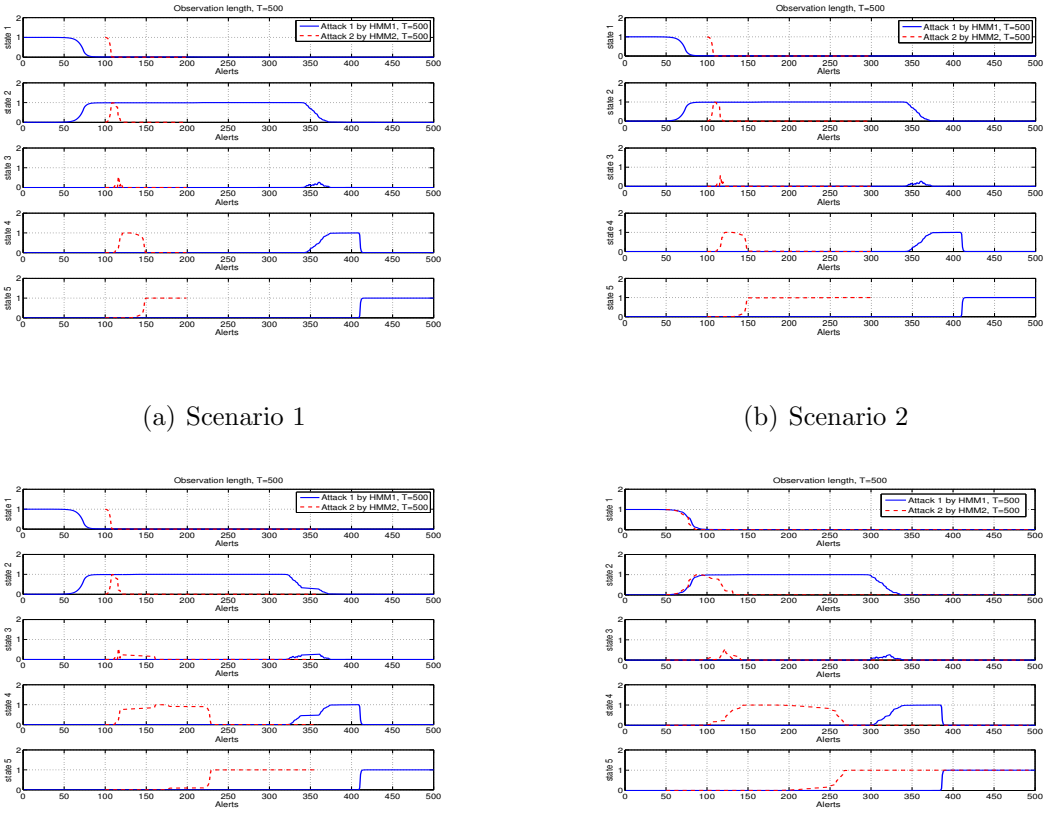
(c) Scenario 3

(d) Scenario 4

Fig. 4.7.: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 based on R-LRHMM, T=10

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. 4.9. For example, for interleaving Scenario 2, Fig. 4.9(a) provides no estimate for state probability of States 2 and 4 for the first 200 alerts when $\epsilon_2 = 0$, as compared to Fig. 4.9(b) when $\epsilon_2 = 0.001$. A similar observation can be made by comparing Figs. 4.9(c) and 4.9(d) for the first 350 alerts of State 2.

In summary, Figs. 4.5, 4.6, 4.7, and 4.8 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, LRHMM+AD is more robust in terms of



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

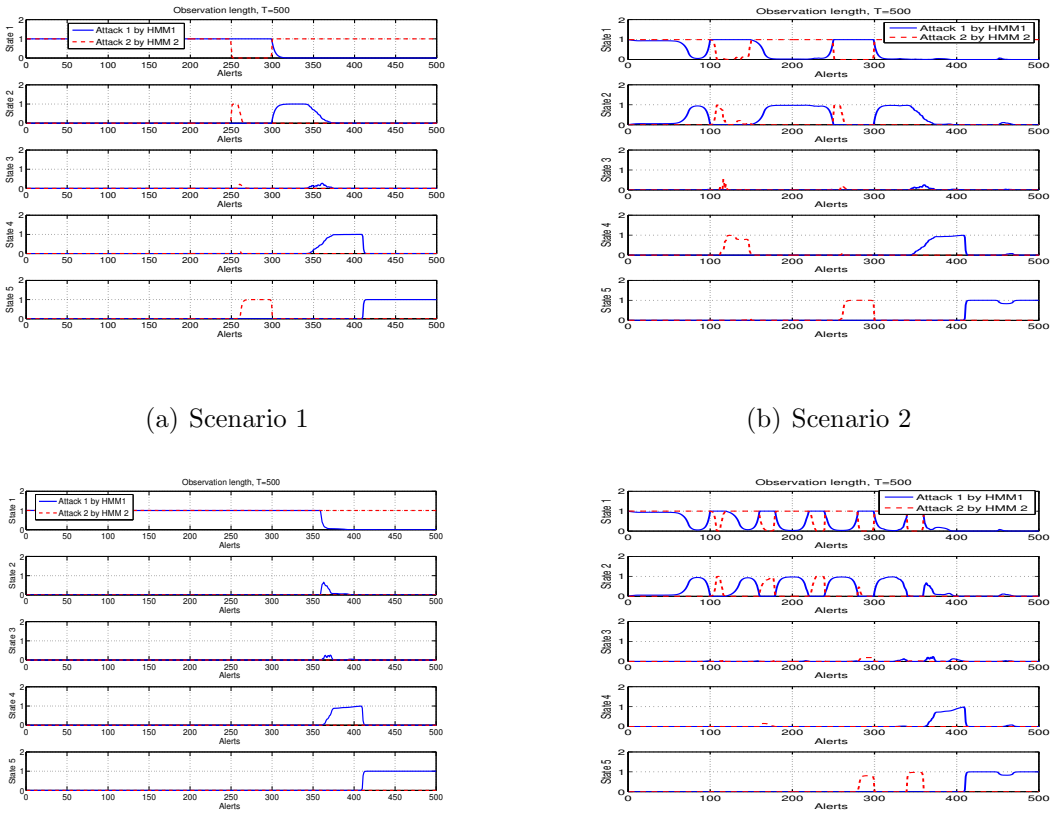
(d) Scenario 4

Fig. 4.8.: State Probability of Attacks 1 and 2 Detected by HMM1 and HMM2 based on LRHMM+AD, T=10

having a better state probability estimation metric at a higher degree of interleaving as compared to R-LRHMM.

4.4.3 Attack Risk Probability

Figs. 4.10 and 4.11 show the attack risk probability, defined in Chapter 3, for both DARPA multi-stage attacks using R-LRHMM and LRHMM+ADR-LRHMM 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



(a) Scenario 1

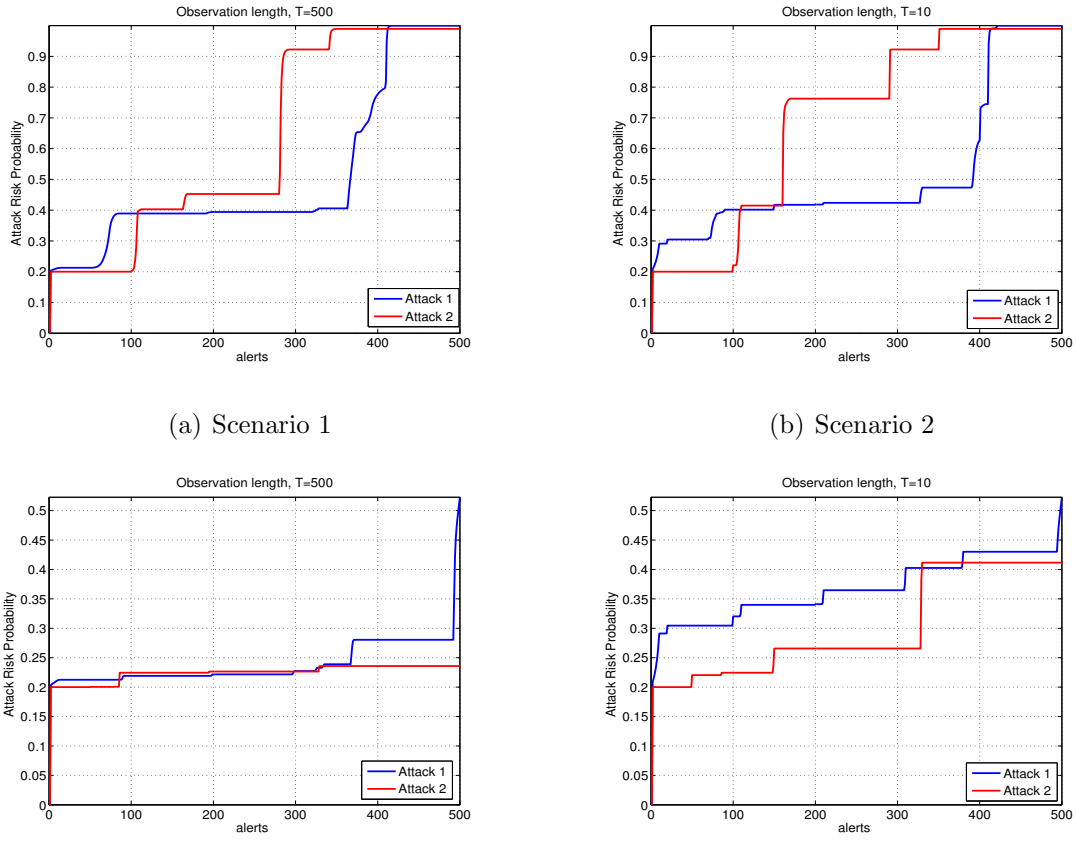
(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 4.9.: Effect of ϵ_2 on State Probability based on R-LRHMM

to detect. Fig. 4.11 shows that LRHMM+AD 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. 4.4(c) and 4.4(d). Note that there is no significant difference is shown between the case of $T = 10$, and $T = 500$. Also Note that after 100 alerts, Attack 2 progresses relatively quickly, and reaches the compromised state well before Attack 1. In contrast, however, R-LRHMM underestimates the progress of Attacks 1 and 2, as shown in Figs. 4.10(c) and 4.10(d), because R-LRHMM fails to detect some of the states for Scenarios 3 and 4, as illustrated in the previous subsection. Fig. 4.10(c) shows that both attacks progress at a slow pace. However, this discrepancy is not true as depicted in Fig. 4.11(c) where LRHMM+AD shows



(a) Scenario 1

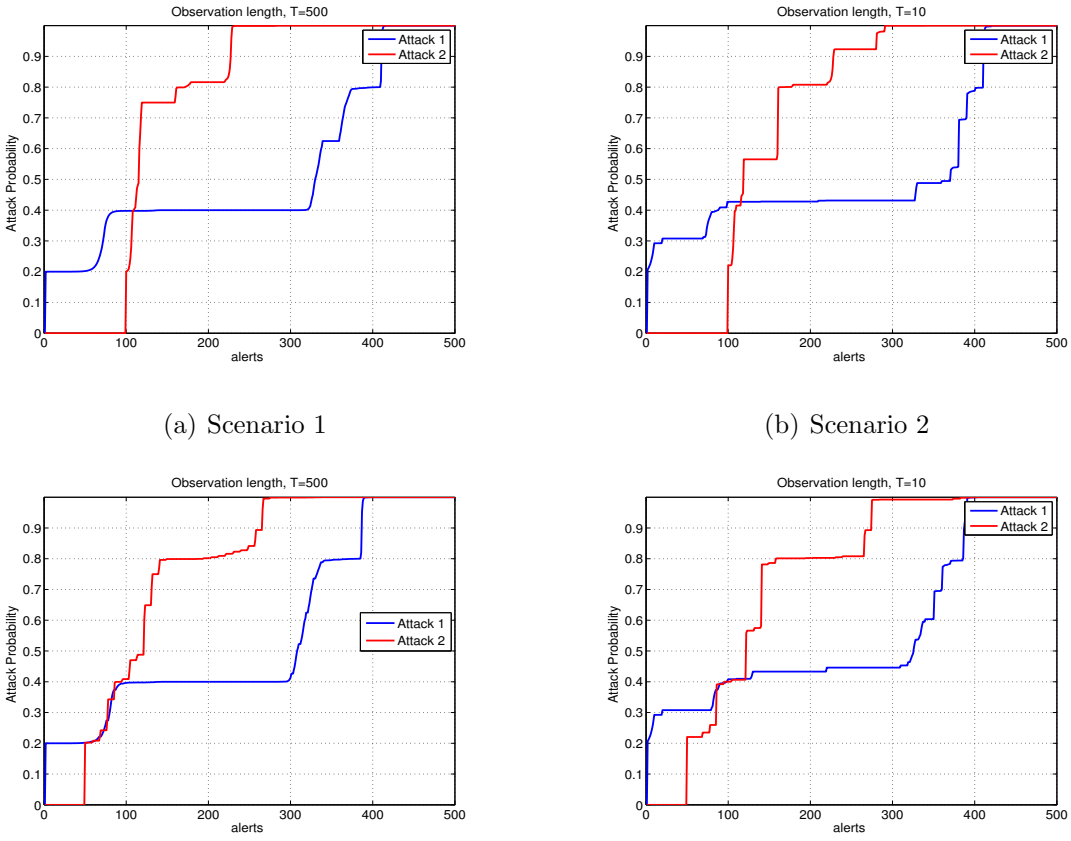
(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 4.10.: Attack Risk Probability Based on R-LRHMM

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. 4.10(d) shows that Attack 1 progresses faster than Attack 2, which is also not true, as depicted in Fig. 4.11(d) which indicates Attack 2 is faster than Attack 1. This inaccurate detection of these attacks can adversely affect response decisions, especially, when a priority-based response mechanism is employed [5].



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

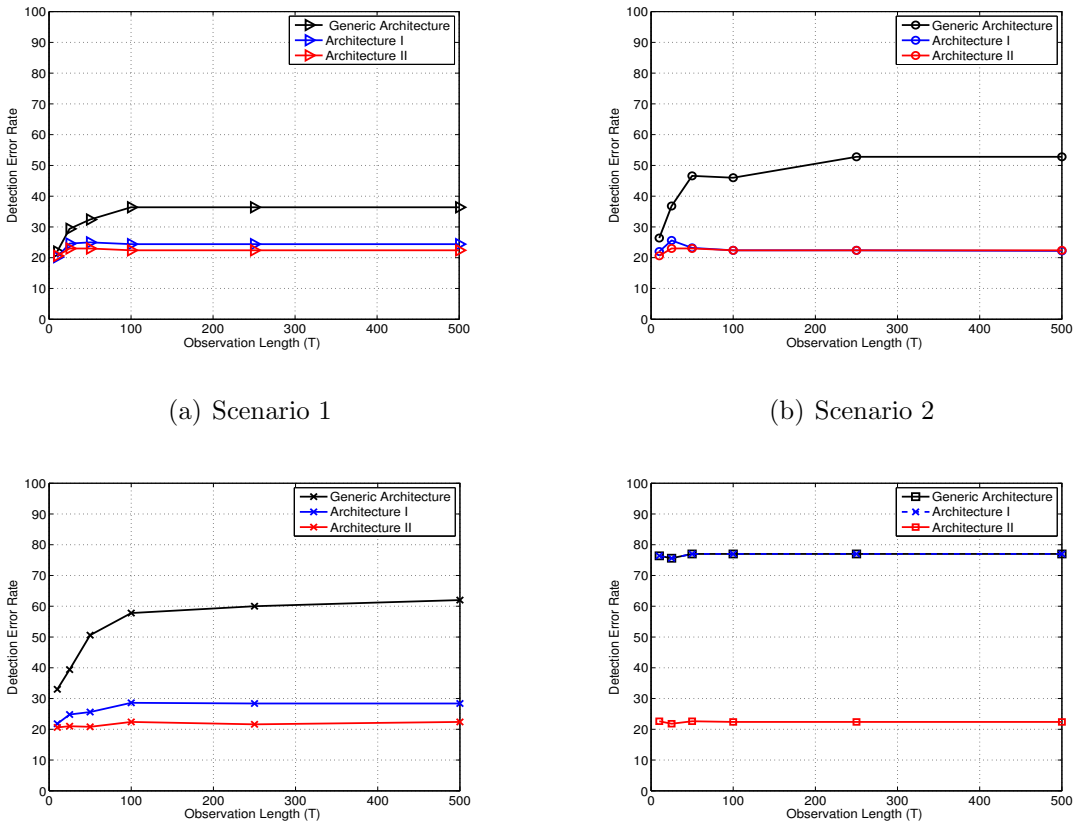
(d) Scenario 4

Fig. 4.11.: Attack Risk Probability Based on LRHMM+AD

4.4.4 Error Rate Performance

The next performance measure we study 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.

Note that 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



(a) Scenario 1

(b) Scenario 2

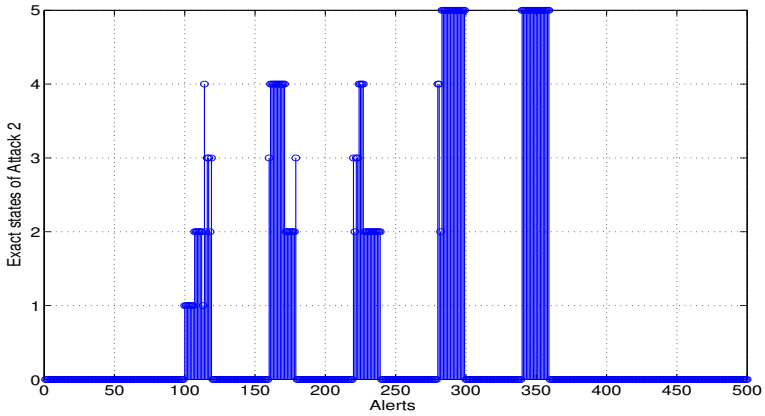
(c) Scenario 3

(d) Scenario 4

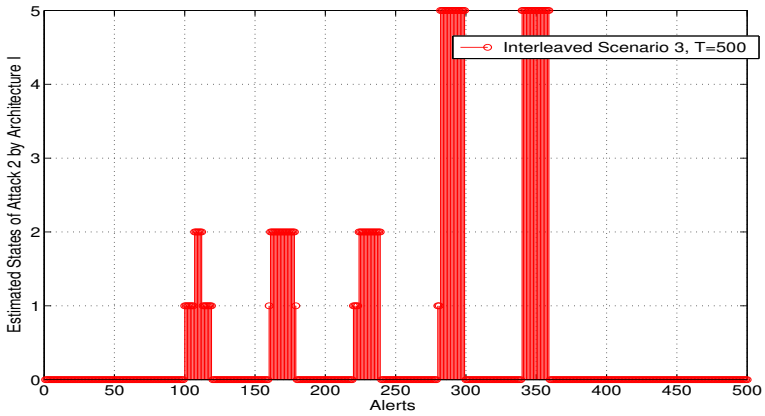
Fig. 4.12.: State Detection Error Rate at Various Interleaving Scenarios

of sequence due to the packets generated by the attacker. In the next chapter, we analyze the effect of False Positives (FPs) and False Negatives (FNs) introduced by the SNORT alert generation system on the state estimation error.

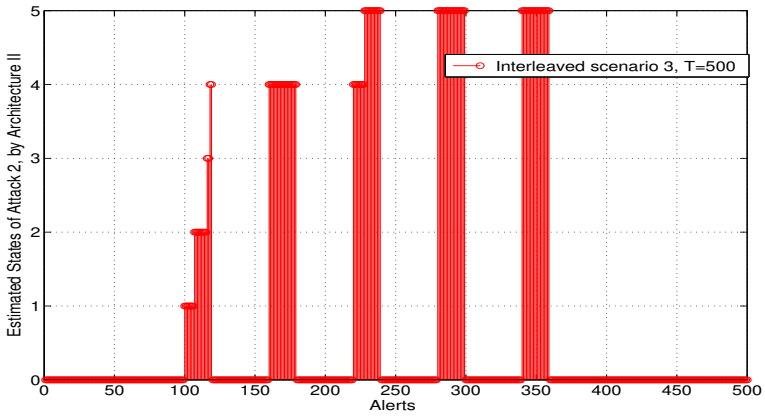
Fig. 4.12 shows the plot of ER for different interleaving scenarios and for several values of T. Note that the error for R-LRHMM is due to the aforementioned reasons (1) and (2), while the error for Architecture 2, is due to only reason (2). It can be seen from the figure that the proposed architectures outperform the generic architecture for interleaving Scenarios 1,2, and 3. However, for Scenario 4, both R-LRHMM and the generic architecture have similar ER, which is higher than LRHMM+AD. It can also be noted that the ER for LRHMM+AD remains almost constant with respect to T



(a) Exact States



(b) R-LRHMM



(c) LRHMM+AD

Fig. 4.13.: Comparison between Architectures I and II in detecting stages of Attack 2 for Scenario 3

Table 4.2.: 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	1	1	1
		Attack 2	1	1	1
	II	Attack 1	4	5	5
		Attack 2	5	5	5

and is also the same for all scenarios. Similarly, R-LRHMM has an almost constant ER with respect to T; however, its ER performance gets worse as the degree of interleaving increases as compared to LRHMM+AD. For instance, for Scenario 4, the ER for R-LRHMM is as high as 77% as compared to LRHMM+AD which has a value of 22%.

Note that 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.

4.4.5 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 4.2 provides the results for three different values of T. It can be observed that LRHMM+AD outperforms R-LRHMM 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 also be seen in Figs. 4.5(b), 4.6(b), 4.7(b), and 4.8(b).

For Scenarios 3 and 4, LRHMM+AD detects more stages than R-LRHMM. For example, all five stages of Attack 2 are detected in Scenario 3 using LRHMM+AD for $T = 500$, while R-LRHMM only detects Stages 1 and 5. Fig. 4.13 shows the estimated states by HMM2 for Attack 2 using Scenario 3. It can be observed that with R-LRHMM, Stages 3 and 4 are not detected. The advantage of LRHMM+AD is more apparent in Table 4.2 when the interleaving Scenario 4 is used.

Fig. 4.13 and Table 4.2 show the importance of this performance metric in terms of how much lead time is available to respond to ongoing attacks. For instance, Fig. 4.13(b) shows that R-LRHMM detects State 2 then immediately detects State 5 of Attack 2, which implies that not enough lead time is available to respond to a progressing attack. While Fig. 4.13(c) shows, on the other hand, that LRHMM+AD detects all states of Attack 2 in a correct sequence similar to the synthesized input states of Attack 2 shown in Fig. 4.13(a). This performance metric establishes the importance of considering a large number of states in modeling the HMM in essence

that the effect of missing a few number of states does not drastically impact lead time while making real time response decisions to an ongoing multi-stage attack.

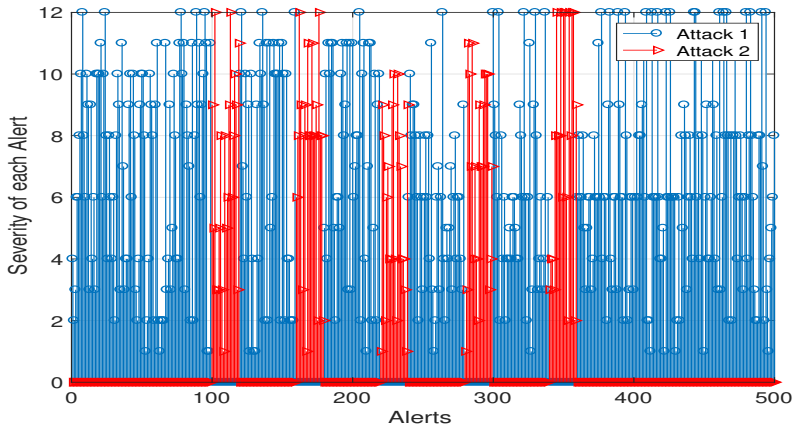
4.4.6 Performance Evaluation Using Synthesized Datasets - Case Study 2

The evaluation experiments in the previous case study 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 that 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 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.

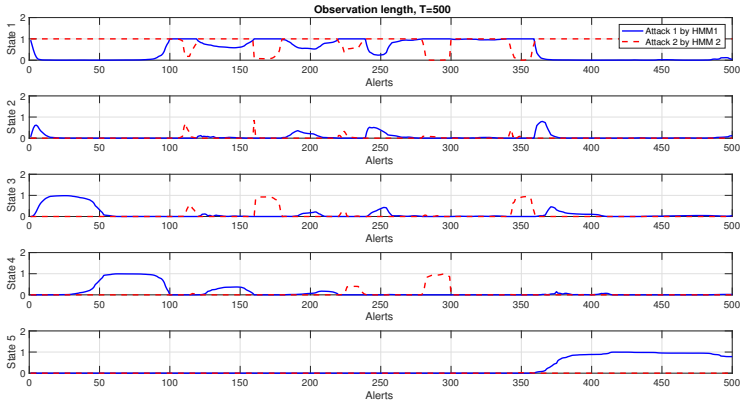
In particular, an HMM generator [53], which generates sequences for HMM, is used to orchestrate several instances of a multi-stage attack type. In this case study, the original DARPA2000 dataset is used for training, and the generated synthesized datasets is used for testing.

Performance Evaluation - Two Synthesized Multi-stage Attacks

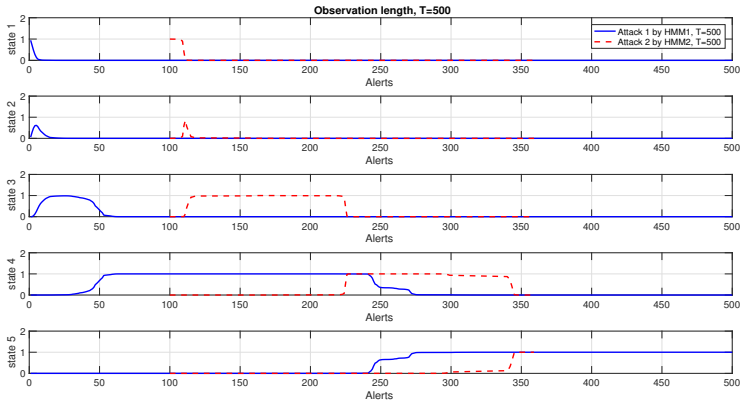
Fig. 4.14 shows the state probability of synthesized Attacks 1 and 2 for the interleaving Scenario 3 detected by HMM1 and HMM2 for R-LRHMM, Fig. 4.14(b),



(a) Exact States



(b) R-LRHMM



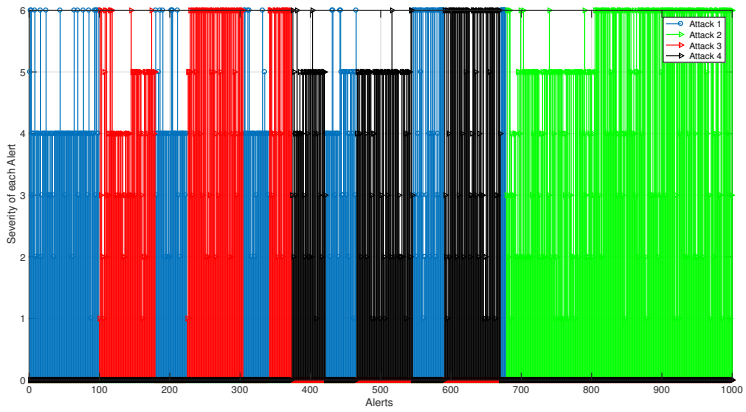
(c) LRHMM+AD

Fig. 4.14.: State Probability of Synthesized Attacks 1 and 2 for the Interleaving Scenario 3 Detected by HMM1 and HMM2 Based on both Architectures, T=500

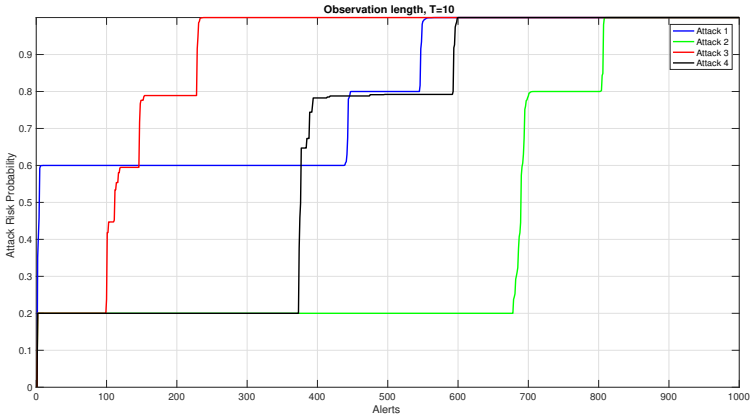
and for LRHMM+AD, (Fig. 4.14(c)). It can be observed in Fig. 4.14 that the results of the case study of synthesized multi-stage attacks are consistent with the previous results for Scenario 3 from Case study 1, discussed in Subsection 4.5.2 (Figs. 4.4(c), 4.5(c) , 4.6(c)), in terms of detection performance and state estimation. In particular, LRHMM+AD detects all stages of the interleaved multi-stage attack scenario. In contrast, R-LRHMM 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.

Performance Evaluation - Four Synthesized Multi-stage Attacks

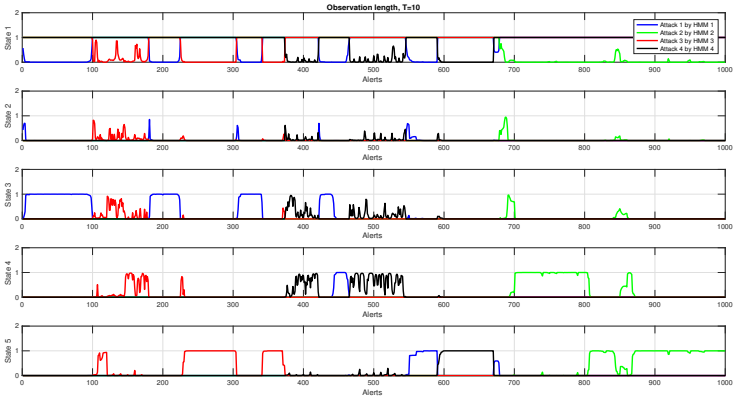
The proposed architectures can be applied to more than two simultaneous attacks, as well as attacks with different instances. We have conducted several experiments, using synthetic datasets, to study the effect of having more than two simultaneous multi-stage attacks on the performance of the proposed architectures. In particular, LRHMM+AD performs better than R-LRHMM 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. R-LRHMM 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. 4.15 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. 4.15(b) shows that the proposed architecture estimates the progress of the attack



(a) Exact States



(b) R-LRHMM



(c) LRHMM+AD

Fig. 4.15.: State Probability of Synthesized Attacks 1 - 4 Detected by HMM1 and HMM2 Based on R-LRHMM, T=10

correctly. Due to page limit, we show the results for only one scenario of interleaving from the four multi-stage attacks.

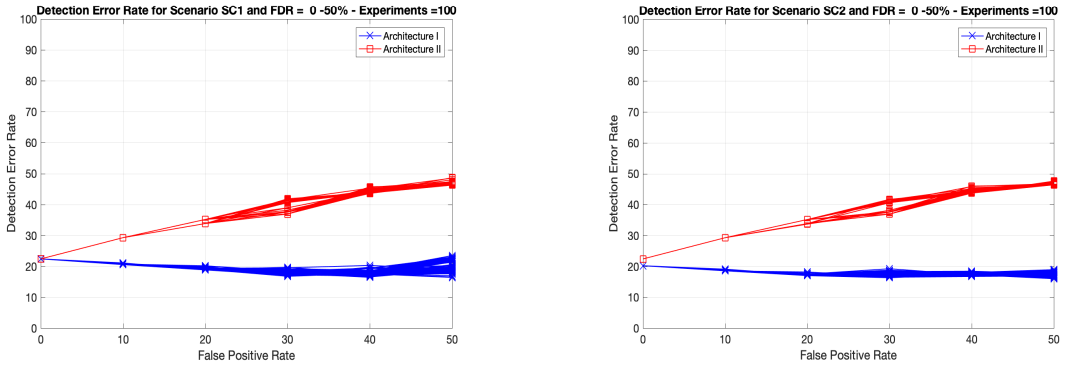
4.4.7 Performance Evaluation - Impact of False Positives (FPs) and False Negatives (FNs)

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 [10]. 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 that 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 all the alerts generated by the SNORT.

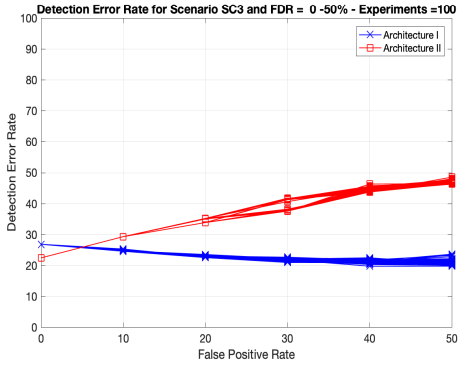
The results for the impact of FDR for both architectures are shown in Fig. 4.16. It can be noticed that the performance of LRHMM+AD degrades with the increase of FDR. This lowered performance is expected as some of FPs are also "demultiplexed" and affect the TPs in their respective substreams when each of these substreams is processed by the associated HMM template.

However, for R-LRHMM, the general trend observed is an improvement in the detection error rate performance which is more noticeable for Scenario 4 (Fig. 4.16(d)).



(a) Scenario 1

(b) Scenario 2

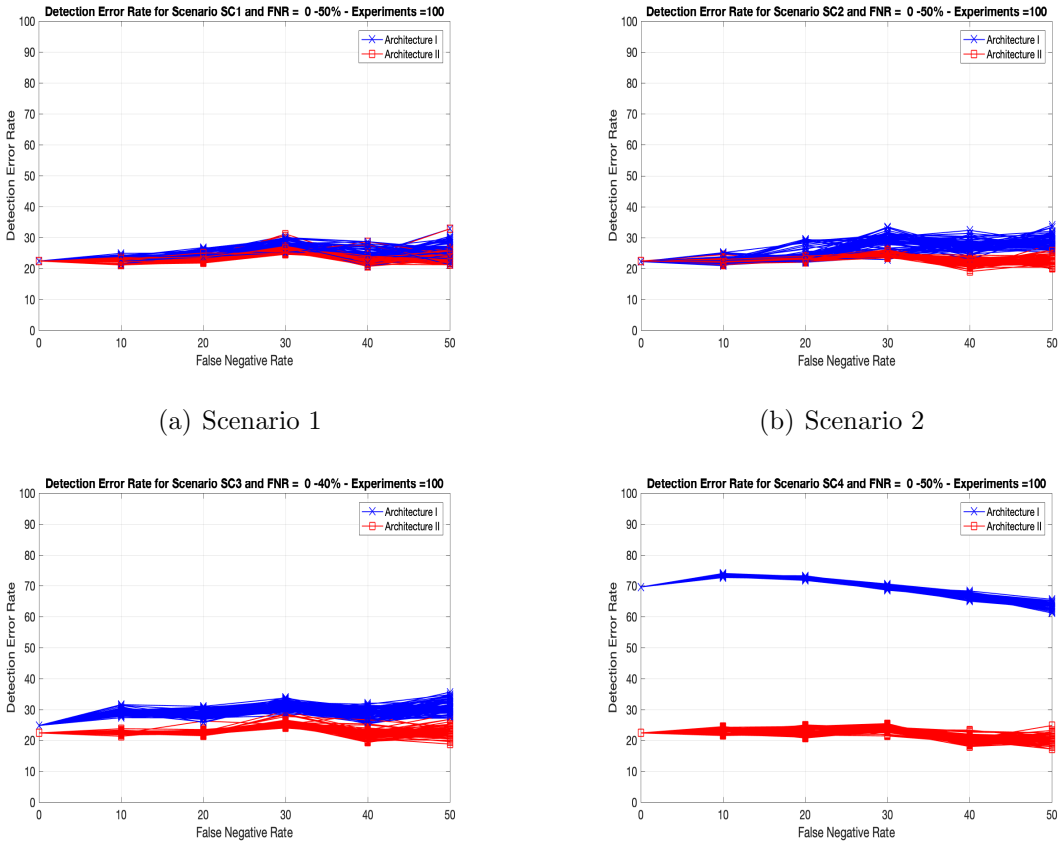


(c) Scenario 3

(d) Scenario 4

Fig. 4.16.: The Impact of False Positives on the State Detection Error Rate of R-LRHMM & LRHMM+AD in Scenarios 1-4 Using Various False Discovery Rates ($FDR = 0\% - 50\%$) - The Observation Window Size = 500 and the Number of Experiments = 100

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 that an HMM template for R-LRHMM always leads to State 1 for unrelated observations. Therefore, as the percentage of FDR increases,



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 4.17.: The Impact of False Negatives on the State Detection Error Rate of R-LRHMM & LRHMM+AD in Scenarios 1-4 Using Various False Negative Rates ($FNR = 0\% - 50\%$) - The Observation Window Size = 500 and the Number of Experiments = 100

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. 4.17. The effect of the FNR on the performance of LRHMM+AD is shown in Fig. 4.17. Note that 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. 4.16 and 4.17.

For R-LRHMM, in addition to reason (2), the reason (1) (mentioned in Subsection 4.4) also comes into play, whereby FNs can eliminate some unrelated alerts and 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.

4.5 Conclusion

This chapter 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, LRHMM+AD, which employs a demultiplexing mechanism, exhibits more robustness and better performance as compared to R-LRHMM. For the performance assessment of these architectures, we use 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.

As mentioned earlier, the performance of LRHMM+AD degrades with the increase of FDR. This lowered performance is because 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. The limitation of R-LRHMM regarding the high degree of interleaving, in addition to the degradation of the performance of LRHMM+AD in the presence of high volume of FPs and FNs, motivate the need for LRHMM+ADI which will be discussed in Chapter 5.

5. HIGHLY RESILIENT ERGODIC HIDDEN MARKOV MODEL ARCHITECTURE (HR-EHMM)

As mentioned in Chapter 4, the performance of LRHMM+AD degrades with the increase of FDR. This lowered performance is because some of FPs are also "de-multiplexed" and affect the TPs in their respective substreams when each of these substreams is processed by the associated HMM template. The limitation of R-LRHMM regarding the high degree of interleaving, in addition to the degradation of the performance of LRHMM+AD in the presence of high volume of FPs and FNs, motivate the need for a new architecture.

In this chapter, we propose HR-EHMM, an HMM-based detection architecture with a backtracking feature, which can detect the occurrence of multiple multi-stage attacks and provide insights about the dynamics of these attacks in the presence of false alarms. We quantitatively evaluate the detection accuracy of HR-EHMM using multiple simulated multi-stage attack scenarios where we control the behavior of the IDS (SNORT) to study the effect of false and missing alerts. In other words, we vary the behavior of the attacker, in terms of stealthiness, and vary the behavior of SNORT, in terms of false and missing alarms, to demonstrate the efficacy of HR-EHMM.

5.1 Introduction

One of the most essential requirements for having high-assurance Cyber systems is to develop advanced and sophisticated attack detection and prediction systems [5]. Most detection systems have the capability to detect a single-stage attack or to detect each of the stages of a multi-stage attack independently. However, due to the inability to analyze the chain of the attack activity as a whole, the detection of multi-stage attacks poses a daunting challenge to the existing intrusion detection methods. This

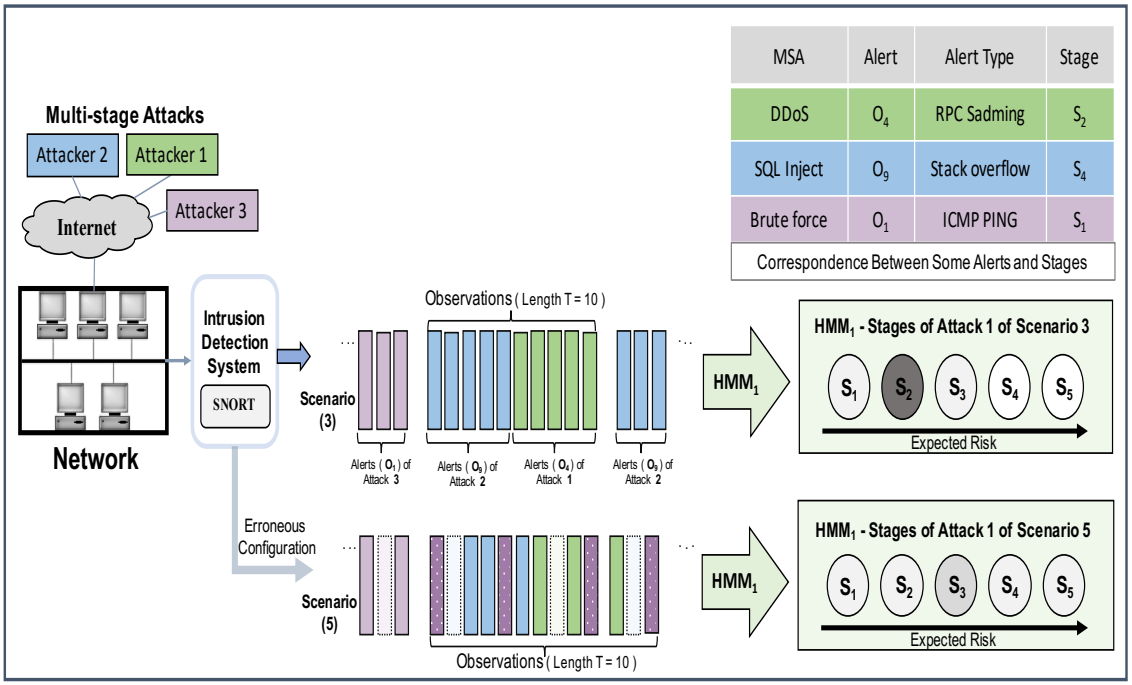


Fig. 5.1.: State Estimation of Interleaved multi-stage attacks Scenarios at Time t

challenge is exacerbated if multiple multi-stage attacks are interleaved. The difficulty in detecting interleaved multi-stage attacks comes from unrelated observations resulting from unrelated attacks that conceal the detail of activity chains of multi-stage attacks. To elaborate on this challenge, Fig. 5.1 exhibits three multi-stage attacks interleaved in five possible scenarios. The correspondence between the Alert type and stages for some observations of these multi-stage attacks is shown in the top-right table. A hypothetical real-time state estimation of Attack 1 for scenarios (3), (4) and (5), assuming that Attack 1 has five stages, is shown on the right side of the figure. A darker state represents a higher probability for the HMM state estimation (i.e., indicates a higher degree of certainty about the current state). Fig. 5.1 shows an example of the impact of interleaving among alerts of three multi-stage attacks on the performance of an HMM state estimation. For instance, in Fig. 5.1, we assume an erroneous configuration for the IDS in scenario (5) that can cause a high rate of false and missing alarms. Hence, due to both reasons, the interleaving of multi-stage

attacks and the erroneous configuration of the IDS, the uncertainty about the current state for Attack 1 in scenario (5) can be higher than in scenario (3).

5.2 Highly Resilient Ergodic Hidden Markov Model Architecture

In this section, we propose HR-EHMM to detect the aforementioned attack scenarios. Note that for HMM structure, we focus on the stream of alerts generated by the IDS, which may consist of True Positive (TP) and False Positive (FP) observations belonging to multiple multi-stage attacks. The design of the HR-EHMM, depicted in Fig. 5.2, is based on modifying HMM_k template parameters (λ_k) (i.e., matrices A_k and B_k) to deal with the interleaved alerts, as discussed next.

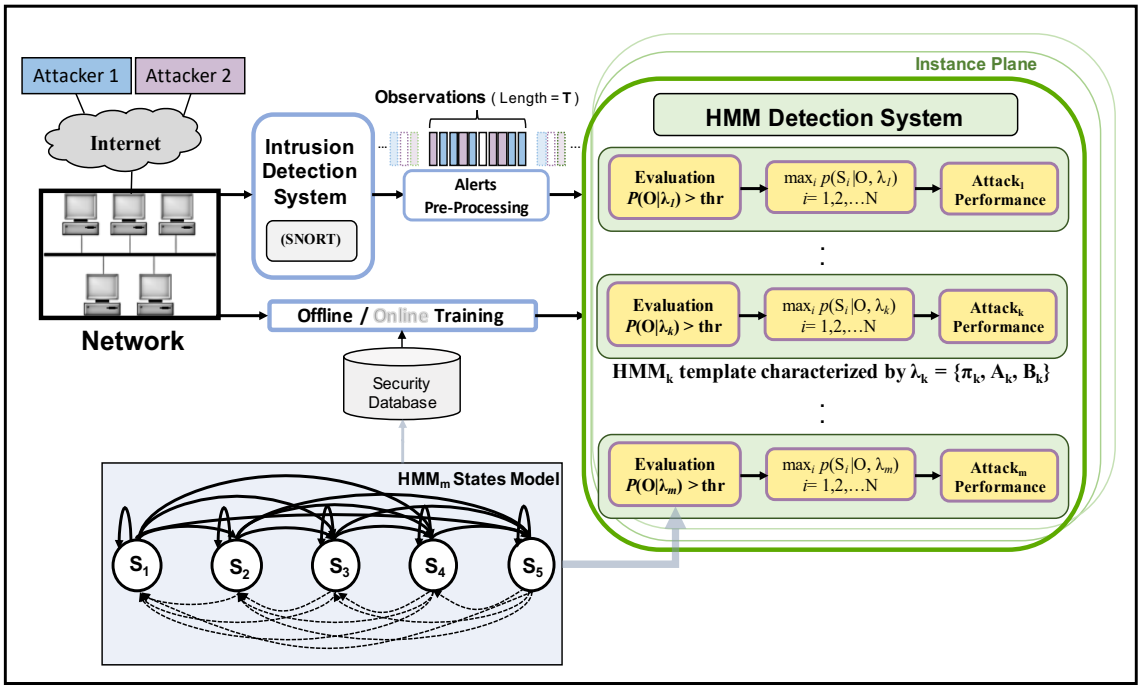


Fig. 5.2.: The Proposed Architecture (HR-EHMM) for Multiple multi-stage attacks Detection using a set of HMM Templates

The HMM detection system processes alerts in windows of length T (i.e., observation sequence $O = \{o_1, o_2, \ldots, o_t, \ldots, o_T\}$). Among T observations, there are L_k observations belonging to Attack k, and the remaining $T - L_k$ observations are

considered by the HMM$_k$ as interfering (unrelated) alerts. For HMM$_k$, we consider the current state as the most likely state that can be inferred by observing $T - L_k$ unrelated observations. In other words, the occurrence of an unrelated observation o_t leads, with high probability, to a nulling transition. This can be done by adding transition probability (ϵ_1) in the right-left transition of the A_k matrix. Hence, instead of introducing a new state to the model, we let all other states return back to a previous state with a very low probability whenever alerts from unrelated attacks occur. Subsequently, the matrix A_k becomes as follows:

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N_k} \\ \epsilon_1 & a_{22} & \cdots & a_{2N_k} \\ \epsilon_1 & \epsilon_1 & \cdots & a_{3N_k} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_1 & \epsilon_1 & \cdots & a_{N_kN_k} \end{bmatrix}$$

Note that in our experiments, we choose $\epsilon_1 = 1 \times 10^{-6}$, which is a small value that does not significantly affect the transition probability matrices, A_k, obtained from training. Note also that $\epsilon_1 = 0$ represents the case of generic architecture (G-Arch), which will be used later in the performance evaluation section, for which returning to the previous states in the model is not allowed when HMM$_k$ receives unrelated alerts belonging to other interleaving attacks (i.e., a strict left-right model).

A second modification is needed for the emission probability matrix (B_k) to be most likely to transition to the current state. This can be obtained by introducing an extra column in the emission probability matrix, B_k, to account for an unrelated observation o_t, as follows:

$$B_k = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M_k} & \epsilon_2 \\ b_{21} & b_{22} & \cdots & b_{2M_k} & \epsilon_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{N_k1} & b_{N_k2} & \cdots & b_{N_kM_k} & \epsilon_2 \end{bmatrix}$$

Hence, unrelated observation o_t, occurs with probability ϵ_2, which has a very small value. In addition, setting the probability to ϵ_2 in the last column increases the

probability that observing o_t leads to a nulling transition whenever HMM_k observes the $T - L_k$ alerts from attacks other than Attack k. An important advantage of modeling unrelated alerts in this way is that it simplifies the training of each HMM.

In order to determine whether Attack k is active or not, the evaluation module is utilized as shown in Fig. 5.2. The evaluation probability can be computed using the forward algorithm and compared with a threshold value thr [52]. Note that thr is used to prevent unnecessary computations of the Viterbi algorithm due to inactive attacks. In this chapter, we take a conservative approach by choosing $thr = 0$. In case Attack k is active, HMM_k uses the Viterbi algorithm to decode the most probable hidden states corresponding to a given observation sequence O, as follows [52]:

$$\begin{aligned} x_t &= \max_{1 \le i \le N_k} \gamma_t(i) \\ \gamma_t(i) &= Pr(x_t = s_i | O, \lambda_k) \\ t &= 1, \ldots, T \end{aligned} \tag{5.1}$$

where $\gamma_t(i)$ represents the probability of being in state s_i at time t for a given O. Each HMM template in HR-EHMM depicted in Fig. 5.2 runs the Viterbi algorithm to determine the best hidden state sequence, $X = \{x_1, \ldots, x_t, \ldots, x_T\}$. For a given observation sequence O, the Viterbi algorithm identifies the highest probability along a single path for every o_t $(t \le T)$ and its corresponding state s_i. Accordingly, the algorithm finds the rest of the state sequence by using induction [52].

5.3 Performance Evaluation

In this section, we evaluate the detection performance of HR-EHMM architecture using two scenarios of multi-stage attacks generated by varying the degree of interleaving between attacks. In addition, we study the impact of False Positives (FPs) and False Negatives (FNs) on the performance of HR-EHMM and compare this performance with the generic architecture G-Arch. The reason for using G-Arch for the comparison is that no evaluation has been done in the past for multiple multi-stage attacks [49].

We discuss the experimental results based on the DARPA2000 public dataset [8]. Note that due to the limitation of the DARPA2000 dataset in terms of the number of scenarios and due to the lack of availability of datasets with a large number of attacks, we conduct our experiments based on a synthesized dataset generated from the DARPA2000 dataset. In our experiments, we alter the timestamp of some of the alerts resulting from multiple multi-stage attacks. Accordingly, we can generate a single sequence of alerts that is composed of a mix of attacks without altering the temporal order of the original alerts. Further, some of the IP addresses of the targeted hosts are also modified. The reason for this modification is to simulate multiple simultaneous attacks.

The DARPA2000 dataset contains two DDoS multi-stage attacks labeled as LLDDOS 1.0 and LLDDOS 2.0.2. Both multi-stage attacks have five stages, which can be summarized as IP sweeping, Sadmind probing, Sadmind exploit, DDoS software installation, and Launching. Hence, the HMM model used in HR-EHMM architecture for each multi-stage attack is a five-state model ($N_k = 5, k = 1, 2$), shown in Fig. 5.2, which is trained offline using the BW algorithm [52]. We use SNORT IDS in our experiments and with various rule configurations to process the raw network packets and to generate alerts [9] [10]. The generated alerts are clustered into 12 distinct symbols for each multi-stage attack, i.e., $M_k = 12, k = 1, 2$. These alerts are assigned severity levels by the preprocessing module based on their relations to stages of the multi-stage attack (i.e., alerts with severity 1-4, 5-7, 8-9, 10-11, and 12 correspond to State 1, 2, 3, 4, and 5 respectively). In the next subsections, the performance evaluation and results are discussed.

5.3.1 State Estimation and the Effect of Interleaving

In this subsection, we present the estimation of state probability, $\gamma_t(i)$, from (5.1) for $i = 1, \ldots, 5$ for two different scenarios, SC1 and SC2, with the observation length $T = 500$.

Figs. 5.3(a) and 5.4(a) show the input alerts and severities of both DARPA2000 multi-stage attacks, Attack 1 and Attack 2, for interleaving scenarios SC1 and SC2 respectively. Note that the degree of interleaving in scenario SC2 is higher than in scenario SC1, which indicates a more complexity of attacks and sophistication of actions. Since Attack 2 takes a shorter time to compromise the target and launch DDoS, we manipulate timestamps of Attack 2 so that it spreads across different times of Attack 1.

The exact state corresponding to every incoming alert is considered based on the knowledge of the input alerts and their corresponding states mentioned earlier. Note that errors that might degrade the detection performance of an HMM-based architecture are due to the inconsistency between the type of an alert and the corresponding estimated state. Such inconsistency can arise due to two main plausible reasons:

1. The existence of interfering alerts due to interleaving.

2. The state estimation error resulting from the enforcement of the left-right HMM model along with some of the observations that may be out of sequence due to the packets generated from the irregular behavior of the attacker. Note that this model is stricter in G-Arch than in HR-EHMM due to introducing ϵ_1 in matrix A_k, as discussed in Section II.

It can be seen from Figs. 5.3(b) and 5.4(b) that HR-EHMM can estimate the states of both attacks with a high probability for both scenarios, especially for States 1, 2, 4, and 5. Note that the state probability for State 3 of Attack 1 is low because there are not enough alerts for this stage produced by SNORT in order to be detected by the trained HMM. On the other hand, Fig. 5.3(c) shows that G-Arch fails to detect States 2-5 of Attack 2 for scenario SC1. For scenario SC2, G-Arch performs poorly, in terms of estimating the correct states of all incoming alerts, as it fails to detect any state for both attacks, as depicted in Fig. 5.4(c). The reason for this poor performance is that, as the degree of interleaving increases between alerts, the more

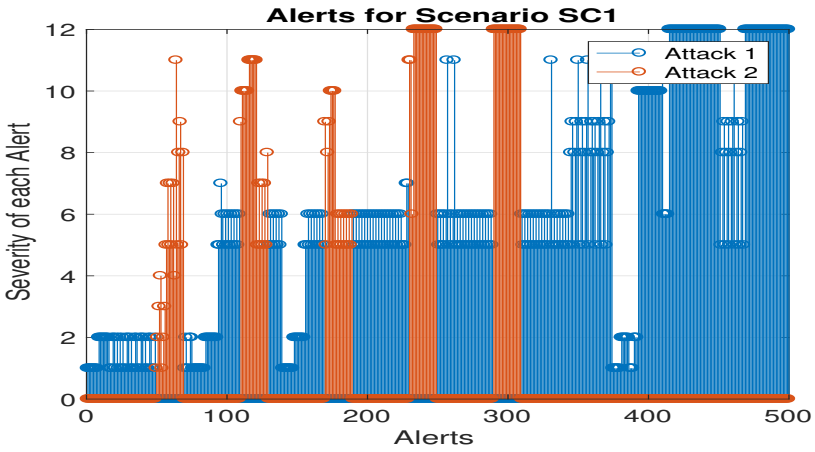
interfering alerts can exist within a given sequence, causing the Viterbi algorithm to incorrectly determine the state probability of the non-interfering alerts.

5.3.2 Detection Accuracy - Impact of FPs and FNs

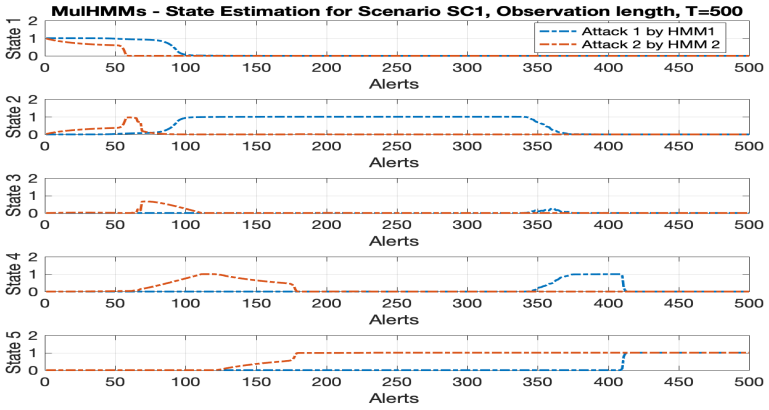
In general, the generated security alarms are noisy and suffer from both FNs and FPs [10]. In the former case, the IDS does not generate alerts for some attack attempts, and in the latter case, the IDS raises false alarms when there are no exploit attempts happening in the network. In this subsection, we analyze the effect of FPs and FNs introduced by the IDS (e.g. SNORT [9]), on the state estimation error. In particular, we use the Detection Accuracy (DA) as a performance measure to study the effect of multi-stage attacks interleaving and the impact of FPs and FNs on both architectures, HR-EHMM and G-Arch. DA is the ratio of the number of correct detected states of the incoming alerts to the total number of alerts. Note that DA = 1 - Error Rate (ER), which discussed in the previous chapters. Formally, DA is given by the following equation:

$$DA = \frac{\text{Number of correct detected states of the incoming TP Alerts}}{\text{Total number of TP Alerts}} \times 100 \quad (5.2)$$

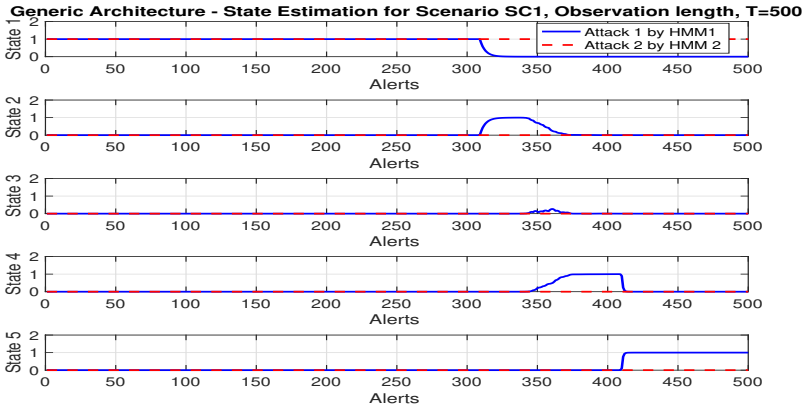
There are two methods to define the false alarm rate: False Positive Rate (FPR), and False Discovery Rate (FDR). On the other hand, True Positive Rate (TPR) is used to define the change in FNs errors. Note that TPR = 1 - False Negative Rate (FNR). In this chapter, we use the FDR-TPR combination to represent the change in the configuration of rules of the IDS (SNORT) and consequently its false detection behavior [10,59]. In particular, several experiments are conducted to study the impact of varying FDR-TPR on the performance of the proposed HMM architecture. In our experiments, we synthesize the dataset by eliminating some of the True Positive alerts (TPs) to mimic FNs, and inject some FPs in a randomized fashion. In addition, due to the randomized elimination and injection of alerts, we conduct 100 experiments for both interleaving scenarios and for each combination of FDR and TPR. We take the



(a) Scenario SC1

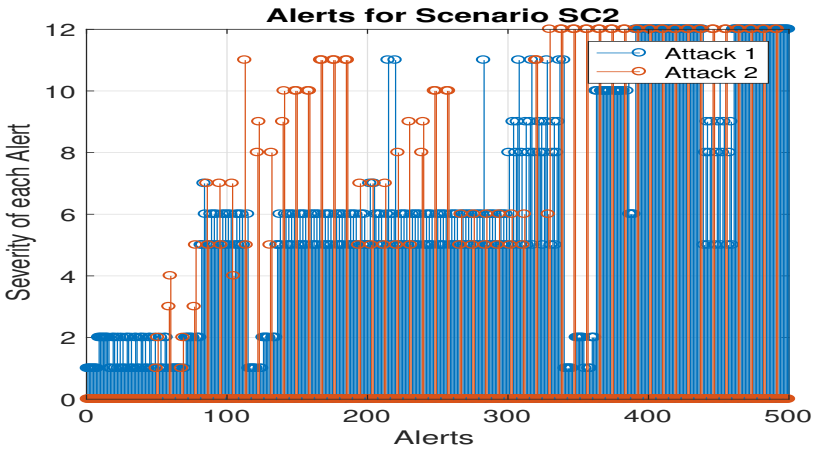


(b) HR-EHMM

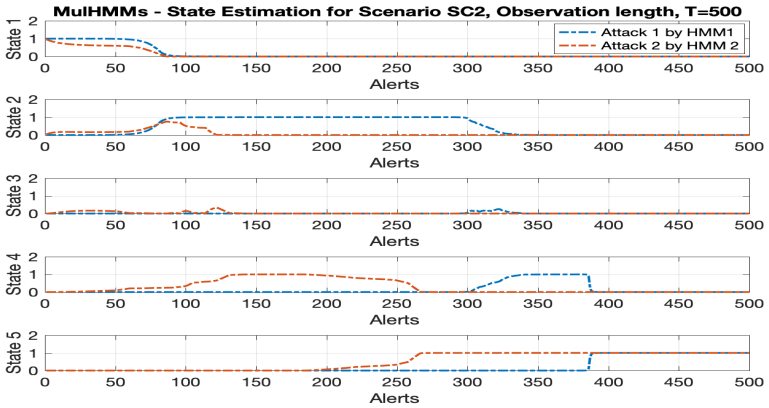


(c) G-Arch

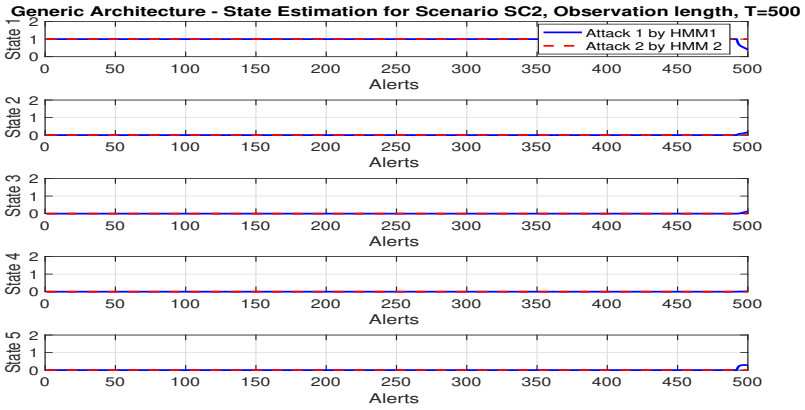
Fig. 5.3.: Interleaved Alerts Scenario SC1 from LLDDOS 1.0 and LLDDOS 2.0.2 Attacks for HR-EHMM and G-Arch



(a) Scenario SC1



(b) HR-EHMM



(c) G-Arch

Fig. 5.4.: Interleaved Alerts Scenario SC2 from LLDDOS 1.0 and LLDDOS 2.0.2 Attacks for HR-EHMM and G-Arch

average over the total number of experiments to identify and filter out any potential outliers.

Note that in our experiments, we study three cases for the distribution of false and missing alarms. In the first case, we assume that the FP and FN errors are uniformly induced by all of the SNORT configuration rules i.e., the impact of FPs errors is uniformly distributed over all the alerts generated by the SNORT. In the second case, we assume that the errors are induced by a subset of the IDS rules and correspond to one state or two consecutive states. In other words, all the errors are localized in one cluster spread over one or two consecutive states. Finally, we consider the case in which errors are generated from a subset of the IDS rules and correspond to two separated states or groups of states (i.e., all the errors are localized in two balanced clusters spread over two non-consecutive states).

Figs. 5.5 and 5.6 show the plots of DA and the impact of varying FDR-TPR for scenarios SC1 and SC2 for these cases. We vary FDR value from 0 to 0.4 and TPR from 1 to 0.6. Note that the point FDR = 0 and TPR = 1 represents the case of having an accurate and precise configuration for the IDS. As mentioned earlier, at this point, FDR-TPR = (0,1), the errors originated from both architectures are due to both aforementioned reasons (1) and (2). Note also, reason (2) causes more errors for G-Arch, in which the enforcement of the left-right model is stricter than in HR-EHMM, which increases the inconsistency between the type of alerts and the corresponding state estimated by the Viterbi algorithm. Hence, the DA for G-Arch is worse than HR-EHMM in each scenario. For instance, for scenario SC1 depicted in Fig. 5.5, the DA value for G-Arch at the point FDR-TPR = (0,1) is as low as 44%, compared to HR-EHMM, which has a value of 78%. Furthermore, the DA degrades for G-Arch as the degree of interleaving increases, as shown in Figs. 5.5 and 5.6, where the DA value for G-Arch at the point FDR-TPR = (0,1) is as low as 23% in scenario SC2, compared to the DA value of 44% in Scenario SC1. The reason for this degradation in the performance of the DA is that, as discussed in Subsection III-A, the number of interfering alerts within an observation sequence increases by increasing the degree of

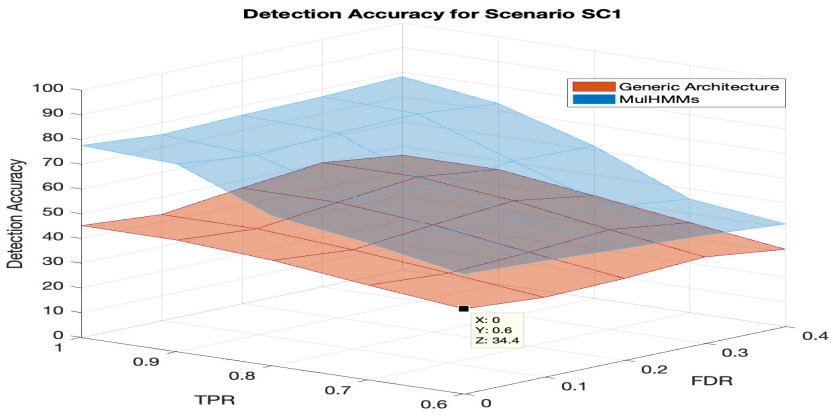
interleaving between alerts, which causes the Viterbi algorithm to estimate the state probability of the non-interfering alerts incorrectly.

It can be noticed from Figs. 5.5 and 5.6 that, in general, as we decrease TPR and increase FDR, HR-EHMM outperforms G-Arch. The effect of varying FPs of the IDS system (SNORT) on the performance of architectures is as follows.

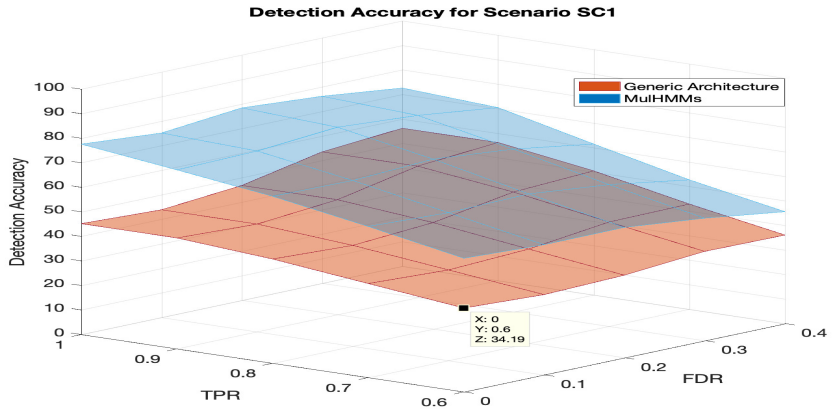
the case of FNs, some of the eliminated TPs by the FNs are accurate alerts (i.e., alerts are correctly sequenced and part of the multi-stage attack), while other TPs that are eliminated by FNs may belong to the irregular behavior of the attacker, which is reason (2) mentioned earlier. The effect on the performance in the case of accurate alerts results in degraded performance of the DA, while for the case of irregular behavior, the DA improves. Such degradation and improvement in the performance can be noticed for different values of TPR, as can be seen in Figs. 5.5 and 5.6.

For instance, in scenario SC1, shown in Fig. 5.5, the DA degrades with lowering TPR for both architectures (i.e., the DA decreases with the increase of FNs). The main reason is that as the TPs decrease, the uncertainty about the current state increases which causes more errors in the state estimation. This problem exacerbates when the number of observations to detect a state in multi-stage attack is low due to the experience of the attacker [49] or due to the behaviors of the attack and the interleaving. For example, the third case of the two clusters depicted in Fig. 5.5(c) has a lower DA than the second case of one cluster in Fig. 5.5(b) when TPR value $= 0.6$ and FDR value $= 0$. A plausible explanation is that the second cluster in the third case eliminates TPs from State 3 and State 4, which have fewer observations than State 1 and State 2. Therefore, the estimation fails to detect correctly some of the states of alerts corresponding to State 3 and State 4.

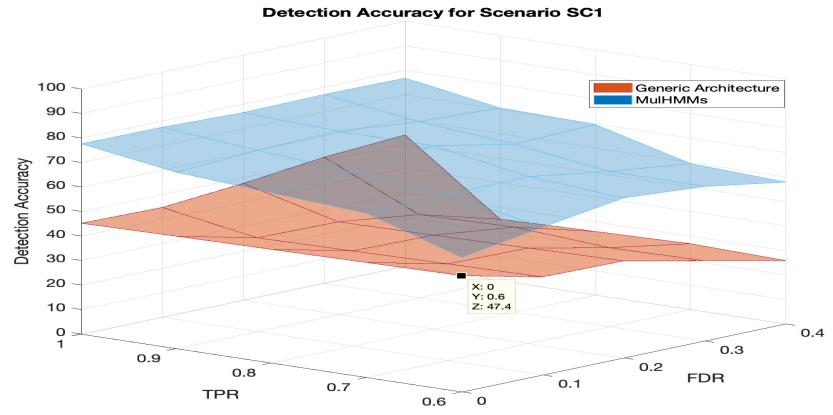
Note that for the case of TPR $= 1$, the DA does not degrade with the increase of FDR in HR-EHMM. The main reason is that we assume a random injection of FPs alerts in all the three cases, with the assumption that two successive FB alerts are most likely not corresponding to the same state. Therefore, under the assumption of



(a) Uniform Distribution of Errors - Scenario SC1

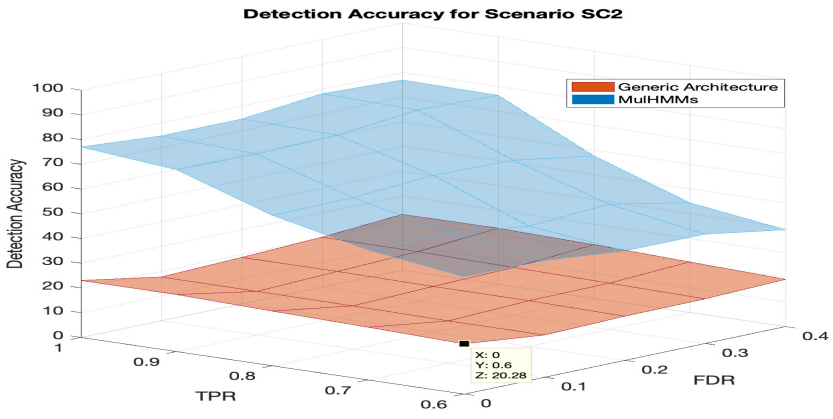


(b) Single Cluster of Errors - Scenario SC1

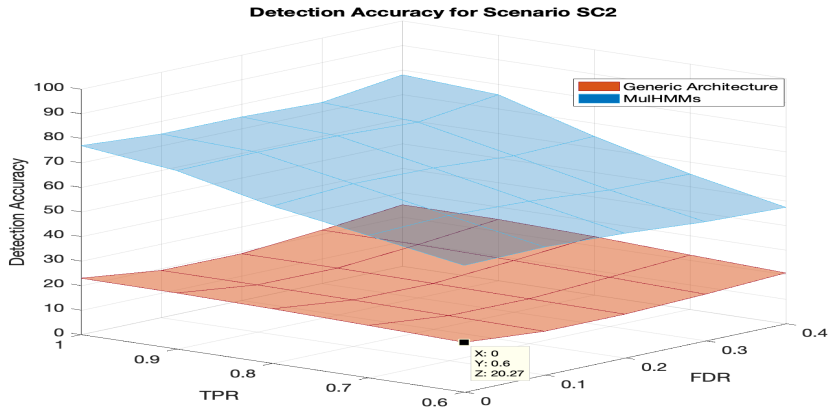


(c) Two Clusters of Errors - Scenario SC1

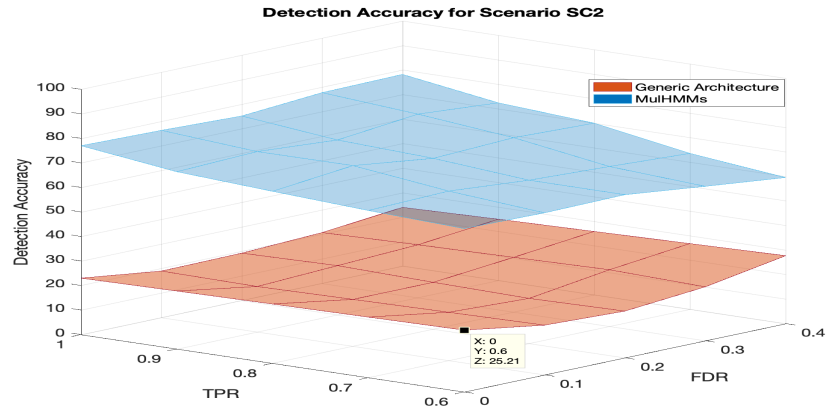
Fig. 5.5.: The Impact of FDR and TPR on the Detection Accuracy for Scenario SC1, Varying the Distribution of FPs and FNs



(a) Uniform Distribution of Errors - Scenario SC2



(b) Single Cluster of Errors - Scenario SC2



(c) Two Clusters of Errors - Scenario SC2

Fig. 5.6.: The Impact of FDR and TPR on the Detection Accuracy for Scenario SC2, Varying the Distribution of FPs and FNs

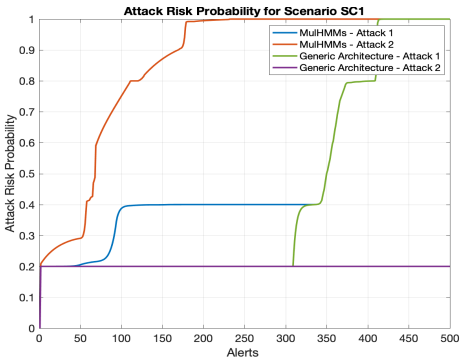
uniform distribution and clustered injection of FPs in the alert stream, the likelihood of staying in the current state increases with the increase in FDR.

5.3.3 Attack Risk Probability

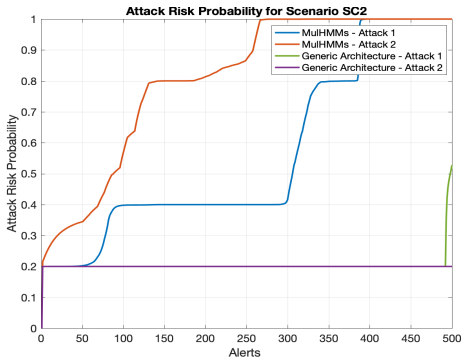
As discussed in Chapter 3, the Attack Risk Probability (ARP) is defined as the probability of how far the current state of an attack is from reaching the final state and causing the maximum damage.

Figs. 5.7(a) and 5.7(b) show the attack risk probability for two interleaving scenarios, SC1 and SC2 respectively, for both HR-EHMM and G-Arch architectures, using DARPA multi-stage attacks, for the case FDR-TPR = (0,1) and for the observation length $T = 500$. It can be noticed that HR-EHMM can accurately track the progress of both attacks for scenarios SC1 and SC2 based on the knowledge of the generated input alerts shown in Figs. 5.3(a) and 5.4(a). Note that after 50 alerts, Attack 2 progresses and reaches the compromise state relatively fast as compared to Attack 1. However, on the other hand, G-Arch underestimates the progress of Attack 1 for scenario SC1 as depicted in Fig. 5.7(a). This is because G-Arch fails to detect State 2 at an earlier point due to the interference from the alerts of Attack 2 as mentioned in the previous subsection.

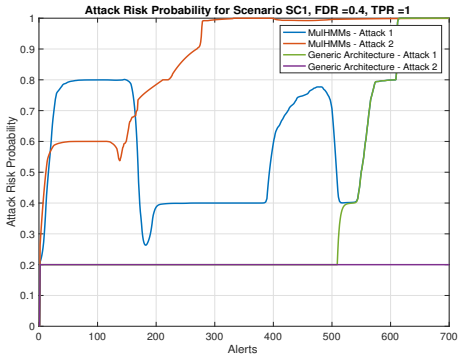
For the case of a non-zero and small value of FDR, FPs and FNs can be distributed across all states. For this case, we observe almost no change in the performance of the system, indicating that short erroneous sequences are tolerated well. However, with a high value of FDR or, in other words, for longer sequences clustered in a specific state, HR-EHMM fails to correctly estimate the true attack risk probability, as shown in Figs. 5.7(c) and 5.7(d). This results in the "oscillatory" shape in two regions corresponding to the two clusters of FP and FN alerts.



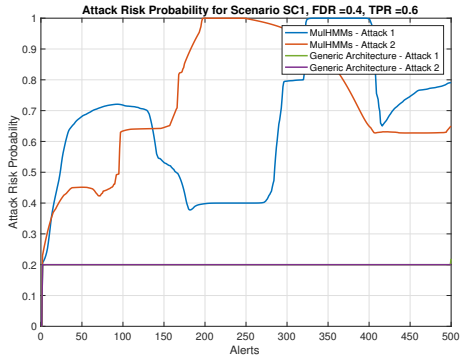
(a) SC1, FDR = 0, TPR = 1



(b) SC2, FDR = 0, TPR = 1



(c) SC1, FDR = 0.4, TPR = 1



(d) SC1, FDR = 0.4, TPR = 0.6

Fig. 5.7.: Attack Risk Probability of Attacks 1 and 2 for SC1 and SC2 and for Various FDR-TPRs for SC1 - Two Clusters of Errors

5.4 Conclusion

In this chapter, we address the detection challenges of interleaving multiple multi-stage attacks targeting a network. We emphasize the importance of this problem by showing how interleaving and stealthy attacks can deceive the detection system. Therefore, we propose an architecture based on an important machine learning technique, the Hidden Markov Model, and demonstrate its efficacy. The architecture, HR-EHMM, can track interleaved multi-stage attacks by estimating the correct states for each incoming alert of the attack. We analyze the impact of the erroneous behav-

ior of IDS on the performance of the proposed HR-EHMM for various interleaving attack scenarios.

6. CONCLUSION AND FUTURE WORK

In Section 6.1 the research contribution is summarized, while in Section 6.2, the future research has been discussed.

6.1 Research Contributions

The main objectives and contributions of this dissertation are as follows. First, we propose a framework for detecting multiple multi-stage attacks occurring consecutively or simultaneously in a network using HMM (sequentially or interleaved). This thesis 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 three architectures based on a well-known machine learning technique, i.e., the Hidden Markov Model, and provide their performance results and computational complexity. The proposed architectures can track interleaved attacks by detecting the correct states of the system for each incoming alert.

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.

The proposed architectures deploy a database of HMM templates of known attacks and exhibit varying performance and complexity. In Chapter 3, we have presented

R-LRHMM architecture. In Chapter 4, we have presented LRHMM+AD architecture [60]. In Chapter 5, we have presented HR-EHMM [61].

6.2 Limitations of the Proposed Architectures

For the research presented in this dissertation, we have made several assumptions regarding the cyber attacks and intrusion detection systems which might need to be overcome in order to deploy the proposed architectures on real-time systems. Some of the limitations and assumptions include the following:

In this research, we have made the assumption that the attacker has no knowledge about the detection architectures in terms of the HMM parameters and the demultiplexing mechanism. However, in general, this assumption may not hold since it is possible for any cyber-based infrastructure to be attacked by an insider that has knowledge about its detection system. Precisely, a knowledgeable attacker can deceive R-LRHMM detection system by interleaving multiple attacks in certain ways (e.g., Scenario 4 discussed in Chapter 3). On the other hand, a knowledgable attacker can deceive LRHMM+AD and HR-EHMM by attacking the demultiplexer component itself (e.g., by using fake IP addresses or by sending a DoS attack on the DeMux). We plan to address this limitation as a future research by using a Moving Target Defense [62] approach or observations filtration techniques.

Another assumption we have made which may not hold is having a database of all possible multi-stage attacks in a network. In practical life, it is quite possible that attacks may not follow the expected behavior constituting, for instance, a zero-day attack, and in this case, the existing HMM templates can not detect all stages of the attack and a new template needs to be generated. One possible solution for the unexpected behaviors issue (e.g., zero-day attacks) is to integrate an anomaly-based detection component along with an online self-adaptive HMM Database to detect such attacks [54].

Furthermore, from memory-efficiency point of view, in R-LRHMM we consider resetting the template to the initial state (State 1) if the corresponding attack becomes idle (e.g., by having another attack). One of the goals of R-LRHMM architecture is to keep as memory-efficient as possible comparing with other multi HMM-based systems. However, since the detection performance for this approach degrades significantly with some scenarios, we plan to study alternative memory-efficient solutions for the proposed architectures that address the tradeoff issue between performance and complexity in HMM-based detection systems.

6.3 Future Work

This work can be extended in various directions. For instance, different variants of designs of HMM can be explored in order to minimize the impact of erroneous IDS. Additional performance-driven design objectives can be investigated such as detection latency and throughput. Also, an HMM-based architecture can be augmented with a real time response mechanism to mitigate the effect of attacks-in-progress before the attacks reach the final stages. In addition, other machine learning techniques such as Recurrent Neural Network (RNN) can be utilized and compared with the proposed model.

6.3.1 Hidden Markov Model Extensions

In this thesis, we have not implemented a real time experimentation in terms of traffic arrival rate. We plan to study the detection latency by evaluating the performance of the proposed architectures using a real-time experimentation. In addition, we can use a temporal-based window approach for the sequence of observations instead of the count-based method.

We plan to study the effect of different alert window sizes and time window sizes on the accuracy and the performance of the HMM prediction model and how to

find criteria for an adaptive window size assignment based on the attack and HMM parameters.

Moreover, we will study the possibility of using a two-layer hierarchical HMM in which the atomic activities and parts of the multi-stage attack are modeled in one layer and parameterized using supervised and unsupervised training and the high-level activities or the key vulnerability for each state are modeled in the other layer and parameterized using CVSS scores [63]. In addition, we plan to study the possibility of using Switching Hidden Semi-Markov Model (S-HSMM) instead of HMM to address the change of state durations and its scalability issues [63].

Furthermore, clustering and pre-processing of alerts can help speeding up the discovery of the attack by decreasing the possibility of any confusion for the HMM due to high degree of interleaving among multi-stage attacks. Specifically, we will investigate utilizing an alert filtration approach to intentionally cause FNs that can lower the high degree of interleaving between MSAs "de-noising effect". This de-noising effect can be noticed in Fig. 5.6(c) where we see a slight improvement in the DA for the G-Arch due to FNs that remove some of the interfering alerts.

6.3.2 Safe Zones - Response

Moreover, the erroneous detection of interleaved MSAs can adversely affect response decisions, especially, when a priority-based response mechanism is employed [5]. We plan to address the challenges of interleaved MSAs along with erroneous IDSs on the existing intrusion response systems such as the recent work [64], which basically employs Partially Observable Markov Decision Process (POMDP) to respond to an ongoing MSA.

Specifically, we plan to develop a cost-based response and recovery strategies that can be used to respond to an ongoing multi-stage attacks and to recover the system's normal or an acceptable operational state once an attack is detected. Such techniques will aim at achieving an acceptable level of resilience in the face of disruptions. In

particular, we plan to develop a novel design concept of formation of Safe Zones (SZs) which is driven by performance metrics such as high level of operational availability and response time. SZs allow isolation of comprised/damaged portion(s) of the CBS, while maintaining high availability of the system functionalities. We plan to model and formulate the problem of finding SZs as a multi-objective optimization problem and to propose heuristics to solve the optimization problem.

6.3.3 Cyber-Physical Systems Application

We plan to study applying Adaptive Threat Management (ATM) mechanisms on a Cyber-Physical Systems (CPS) application. In ATM, we will study not only the detection of MSA, but also risk assessment and response. The main challenges for this research are as follows. The first challenge is how to fuse the information and correlate the alerts from both Cyber and physical IDSs. The second challenge is how to assess the impact on both the Cyber and the physical sub-systems and functionalities/missions for both the MSA and the plausible response actions. We aim to apply ATM mechanism on a Smart Grid application, (specifically in Advanced Metering Infrastructure (AMI)), as an extension for our previous work in [16].

6.3.4 Scalability - Partitioning-based Approach

We plan to study how to find an optimal distribution for detection (sensors), estimation, and controllers (e.g., Firewalls) based on a cost-driven partitioning approach to resolve the scalability and the real-time latency issues.

REFERENCES

REFERENCES

[1] 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.

[2] Wikipedia, "December 2015 ukraine power grid cyberattack," 2015, accessed: 2017-10-21.

[3] CNET, "Wannacry ransomware: Everything you need to know," 2017, accessed: 2017-10-22.

[4] Washingtonpost, "Massive cyberattack hits europe with widespread ransom demands," 2017, accessed: 2017-10-22.

[5] S. Jajodia and M. Albanese, *"An Integrated Framework for Cyber Situation Awareness"*. Springer International Publishing, 2017.

[6] N. Luktarhan, 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, pp. 275–282.

[7] J. Navarro, A. Deruyver, and P. Parrend, "A systematic survey on multi-step attack detection," *Computers & Security*, 2018.

[8] L. L. M. I. of Technology, "Defense advanced research projects agency dataset (darpa)," 2015.

[9] Snort, "The snort intrusion detection system," 2015.

[10] G. C. Tjhai, M. Papadaki, S. M. Furnell, and N. L. Clarke, "Investigating the problem of ids false alarms: An experimental study using snort," in *SEC*, 2008.

[11] 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.

[12] D. of Homeland Security (DHS), "Resilience," 2017, accessed: 2017-10-23.

[13] A. S. Sendi, M. Dagenais, M. Jabbarifar, and M. Couture, "Real time intrusion prediction based on Optimized Alerts with Hidden Markov Model," *Journal of Networks*, vol. 7, no. 2, pp. 311–321, 2012.

[14] Z. Zhang, Z. Peng, and Z. Zhou, "The study of intrusion prediction based on HsMM," *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008*, pp. 1358–1363, 2008.

[15] H. Hu, H. Zhang, Y. Liu, and Y. Wang, "Quantitative Method for Network Security Situation Based on Attack Prediction," vol. 2017, 2017. [Online]. Available: file:///home/durkota/Downloads/3407642.pdf

[16] T. Shawly, J. Liu, N. Burow, S. Bagchi, R. Berthier, and R. B. Bobba, "A risk assessment tool for advanced metering infrastructures," *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 989–994, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/7007777/

[17] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[18] X. Zan, F. Gao, J. Han, and Y. Sun, "A hidden markov model based framework for tracking and predicting of attack intention," *1st International Conference on Multimedia Information Networking and Security, MINES 2009*, vol. 2, pp. 498–501, 2009.

[19] A. R. Hota, A. A. Clements, S. Sundaram, and S. Bagchi, "Optimal and game-theoretic deployment of security investments in interdependent assets," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9996 LNCS, pp. 101–113, 2016.

[20] 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/10.1145/948109.948137

[21] S. Jajodia, N. Park, F. Pierazzi, A. Pugliese, E. Serra, G. I. Simari, and V. S. Subrahmanian, "A Probabilistic Logic of Cyber Deception," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2532–2544, 2017.

[22] K. Haslum, 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, EUROSIM/UKSim2008*, pp. 216–223, 2008.

[23] H. Farhadi, M. Amirhaeri, 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.

[24] 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

[25] M. M. Siraj, H. Hussein, T. Albasheer, and M. M. Din, "Towards Predictive Real-time Multi-sensors Intrusion Alert Correlation Framework," *Indian Journal of Science and Technology ISSN*, vol. 8, no. 12, pp. 974–6846, 2015.

[26] K. Sallhammar, K. Haslum, T. Brekne, M. Elisabeth, G. Moe, and S. J. Knapskog, "Real-Time Risk Assessment with Network," pp. 388–397, 2005.

[27] M. Albanese and S. Jajodia, "A Graphical Model to Assess the Impact of Multi-Step Attacks," 2017.

[28] M. A. Rahman, A. Farooq, A. Datta, and E. Al-Shaer, "Automated synthesis of resiliency configurations for cyber networks," *2016 IEEE Conference on Communications and Network Security, CNS 2016*, pp. 243–251, 2017.

[29] M. A. Rahman and E. Al-Shaer, "Automated Synthesis of Distributed Network Access Controls: A Formal Framework with Refinement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 416–430, 2017.

[30] S. Zonouz and P. Haghani, "Cyber-physical security metric inference in smart grid critical infrastructures based on system administrators' responsive behavior," *Computers and Security*, vol. 39, no. PART B, pp. 190–200, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2013.07.003

[31] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 395–406, 2014.

[32] E. Serra, S. Jajodia, A. Pugliese, A. Rullo, and V. S. Subrahmanian, "Pareto-Optimal Adversarial Defense of Enterprise Systems," *ACM Transactions on Information and System Security*, vol. 17, no. 3, pp. 1–39, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2744298.2699907

[33] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146–169, July 2004.

[34] 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*, vol. 29, no. 7, pp. 1438–1448, 2011.

[35] 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, computer Communications.

[36] 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.

[37] F. Manganiello, M. Marchetti, and M. Colajanni, "Multistep 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. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 101–110.

[38] L. Huiying, P. 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.

[39] C. Onwubiko, *"Situational Awareness in Computer Network Defense: Principles, Methods and Applications: Principles, Methods and Applications"*. IGI Global, 2012.

[40] Z. Zali, M. R. Hashemi, and H. Saidi, "Real-time attack scenario detection via intrusion detection alert correlation," in *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*. IEEE, 2012, pp. 95–102.

[41] F. Alserhani, M. Akhlaq, 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.

[42] 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*, vol. 11, no. 1, pp. 203–218, 2017.

[43] 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 Hawaiian International Conference on System Sciences*, p. 334, 2003.

[44] A. Årnes, K. Sallhammar, 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, pp. 388–397.

[45] 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.

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

[47] 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*. IEEE, 2014, pp. 14–19.

[48] U. S. K. Thanthrige, J. Samarabandu, and X. Wang, "Intrusion alert prediction using a hidden markov model," *arXiv:1610.07276*, 2016.

[49] 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*, vol. 5971, no. c, 2017.

[50] 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

[51] K. Haslum, 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, EUROSIM/UKSim2008*, pp. 216–223, 2008.

[52] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," pp. 257–286, 1989.

[53] M. Inc., "Matlab and statistics/ machine learning toolbox release 2014b, natick, massachusetts, united states." 2018.

[54] 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.

[55] Barnyard2, "Barnyard2: an open source interpreter for snort output files," 2016.

[56] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. New York, NY, USA: ACM, 2002, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/543613.543615

[57] D. Curry, "Intrusion detection message exchange format data model and extensible mark-up language (xml) document type definition," *draft-ietf-idwg-idmef-xml-09. txt*, 2002.

[58] I. I. D. W. Group *et al.*, "Intrusion detection message exchange format," 2002.

[59] A. Beaugnon and P. Chifflier, "Machine learning for computer security detection systems: Practical feedback and solutions," 2018.

[60] T. Shawly, A. Elghariani, J. Kobes, and A. Ghafoor, "Architectures for Detecting Real-time Multiple Multi-stage Network Attacks Using Hidden Markov Model," *Minor Revision ? IEEE Transactions on Dependable and Secure Computing (TDSC)*, Resubmitted August 2019.

[61] T. Shawly, M. Khayat, A. Elghariani, and A. Ghafoor, "Evaluation of HMM-Based Network Intrusion Detection System for Multiple Multi-stage Attacks," *Accepted ? IEEE Network*, 2019.

[62] E. Al-Shaer, *Toward Network Configuration Randomization for Moving Target Defense*, 2011.

[63] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh, "Activity Recognition and Abnormality Detection with," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, Washington, D. C.*, pp. 838–845, 2005.

[64] E. Miehling, M. Rasouli, and D. Teneketzis, "A pomdp approach to the dynamic defense of large-scale cyber networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, Oct 2018.

APPENDICES

A. HMM-BASED GENERIC ARCHITECTURE

Fig. A.1 shows the general architecture of the HMM-based architecture for multi-stage attacks detection. The essential components and tasks in this architecture are as follows:

- Cyber-based System model: this component generates functionality dependency graphs for the whole system and its missions.

- Vulnerability Dependency Graphs (VDG): this component generates VDG templates for all known and possible multi-stage attacks. VDG is created by integrating the functionality dependency graph generated from the cyber-based System model with its vulnerabilities generated from the output of vulnerability scanners.

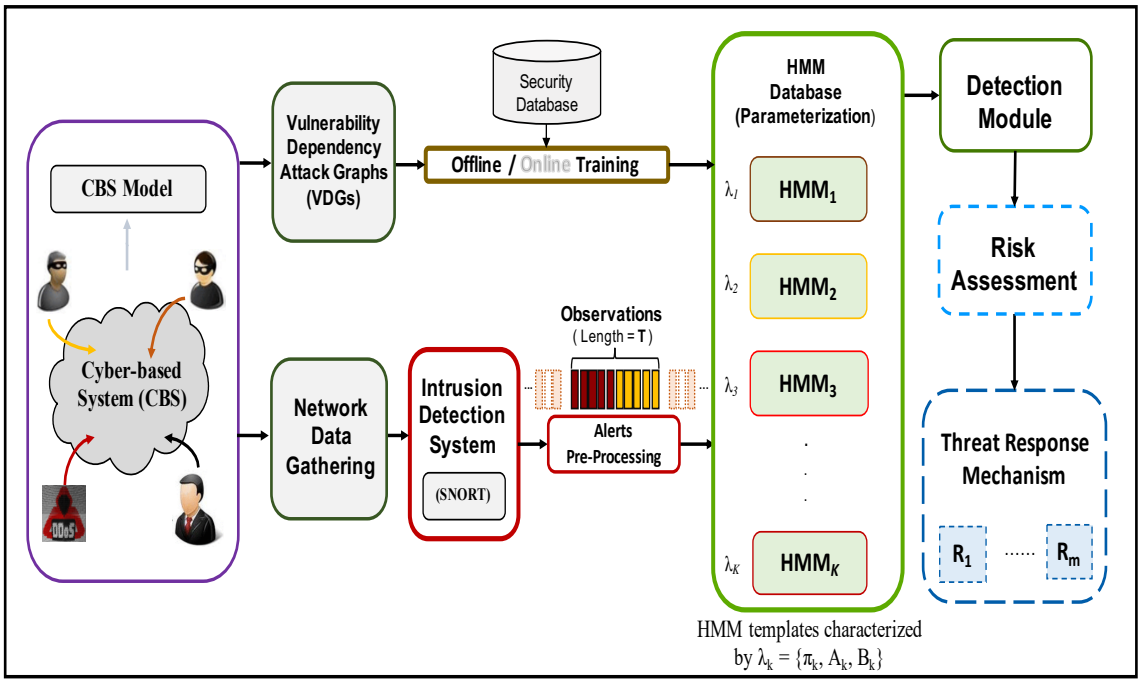


Fig. A.1.: A Generic Architecture for Multiple Multi-stage Attack Detection using an HMM database

- Network Data Gathering: this component extracts necessary information from captured network traffic and activities and prepare it for the detection components.

- Detection: this module tries to detect abnormal activities and generates alerts.

- Security Database: this component stores all Common Vulnerabilities and Exposures (CVE) names, CVSS scores and their computed measures.

- Alerts Clustering and Pre-processing: alerts clustering and Pre-processing component correlates and groups alerts and adjusts their severities to get a better prediction.

- HMM Knowledgebase: this component contains the HMMs parameters obtained by training them with different multi-stage attacks. These parameters include the number of states of the Markov chain, the number of observations and probability matrices.

- Prediction: the prediction component will estimate and probabilistically predict the possibility of current or future cyber problems based on the IDSs alerts observations. This prediction is achieved by applying HMM algorithms to the observations sequence arriving to the prediction component and using an HMM template from HMM knowledgebase component. As a result, we will compute the most probable sequence of states the multi-stage attacks go through and calculate the multi-stage attack probability. This prediction mechanism will be illustrated in more details in the following sections.

- Risk Assessment: this component assess the risk and the impact of the multi-stage attack on the functionalities/missions of the system.

- Response: the response component prepares a set of responses to run on the network based on the result of the prediction and the risk assessment components with a goal to prevent or minimize the damage propagation and to recover the system to a normal mode or to the most possible acceptable functional state.

A.1 Preliminaries

We develop a weighted graph model to represent the topology of the CBS network, known vulnerabilities, and their interdependencies. This model will be used mainly in the risk assessment and response components of the proposed ATM model but we introduce it in this chapter with an example to motivate the need for the prediction mechanism.

The cyber-based system graph model is formally represented as $G = \{V, E\}$, where V represent its components (nodes), i.e., servers, routers, hosts, etc., and E is the set of connections between components. Let $F = \{f_1, f_2, ..., f_m\}$ be the set of functions/services supported by CBS, where a Function Dependency Graph (FDG) f_i is modeled as a directed graph $f_i \subset G$ and $\bigcup_{f \in F} f = G$. Let V_{f_i} (for simplicity V_i) represents the set of components that support the functionality f_i and E_{f_i} (for simplicity E_i) represents the set of edges that support the functionality f_i.

For any $h \in V$, h is a tuple $(id(h), Ser(h), Vul(h), Wei(h))$, where $id(v)$ denotes a unique identification represents the node's IP address, Ser(h) represents the a list of the active services on that node, Vul(h) identifies all the known vulnerabilities on the host h, and Wei(h) identifies the weight of a vertex in f_i which represents its importance and criticality for the functionality f_i which is based on several factors such as the frequency of accessing that component from other hosts and servers. The weight of an edge in f_i identifies its importance for that functionality based on the frequency of using that edge from other hosts and servers to reach and access a critical server.

We model exploits and vulnerabilities using information from various sources such as Snort [9]. Vulnerability Dependency Graph (VDG) Gv_i for a functionality f_i is a directed graph $Gv = (Vuls, Ev)$ where $Vuls$ is the set of vulnerabilities (vertices) and $Ev \subseteq Vuls$ x $Vuls$ is the set of edges. VDG can be generated from Nessus vulnerability scanner output and functionality dependency graph. Intuitively, edges represent causal relationships between vulnerabilities (i.e, an edge from vulnerability v_1 to a

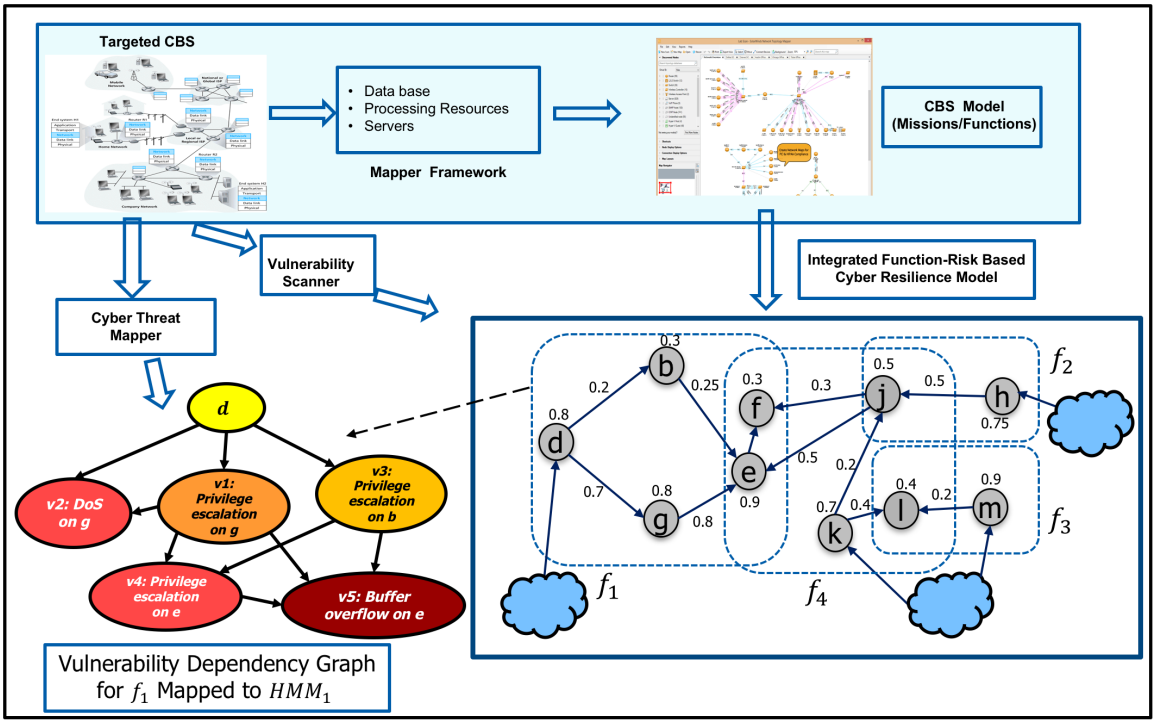


Fig. A.2.: CBS Resilience Model - Example

vulnerability v_2 means that v_2 can be exploited if v_1 is exploited). A vulnerability with zero in-degree can be exploited directly (entry vulnerability that is exploited to initiate attacks). Fig. A.2 shows an example of a CBS Resilience Model with four functions mapped to ten nodes and then generating a VDG for the FDG f_1.

For any $v \in Vuls$, v is a tuple $(id(v), IP(v), p(v), MTTE(v), impact(v))$, where $id(v)$ denotes a unique identification for the vulnerability that comes from Common Vulnerabilities and Exposures (CVE) which is a dictionary of common names (i.e., CVE Identifiers) for publicly known cyber security vulnerabilities [**?**], $IP(v)$ represents the node's id (IP address(es)) that has this vulnerability, p(v) represents the vulnerability exploitability probability, MTTE(v) represents the the mean time to exploit the vulnerability, and impact(v) indicates the vulnerability severity and impact on the functionality. Note that p(v), MTTE(v), and impact(v) can be computed initially using CVSS(v) score.

As shown in Fig. A.2, each VDG will be mapped to an HMM where each vulnerability exploit is represented by a hidden state.

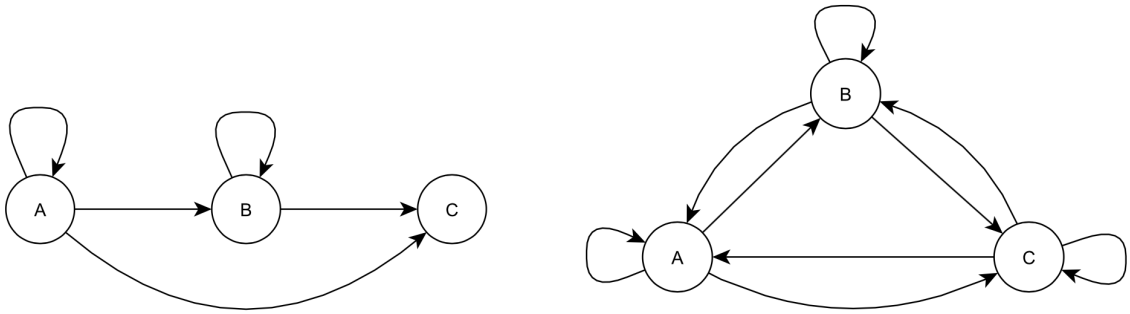
A.2 Correlating Vulnerabilities of Attack Graph with Alerts from HMM

Attack graphs represent prior knowledge about vulnerabilities, their dependencies, and dependency of functions on CBS components. We model an attack graph by enumerating all possible vulnerabilities sequences an attacker can exploit within the CBS to reach his goal (e.g., a critical server), by enumerating all possible attack paths along functional mapping. We assume that an attacker does not relinquish acquired capabilities. Accordingly, dependencies between vulnerabilities can be recorded by attack graphs while keeping attack paths implicitly without losing any information. The CBS component vulnerability model can be automatically created from output of vulnerabilities scanners such as the Nessus vulnerability scanner. We generate HMM-based threat detection templates from attack graphs by mapping vulnerabilities to observation as well as the transition probabilities in HMM. Such mapping will allow generation of HMM-based threat detection templates from attack graphs. Such templates will be maintained in the ATM knowledgebase depicted in Fig. A.1.

A.3 Implementation of ATM - HMM Prediction Model

This section details the application and implementation of the architecture (Fig. A.1) for different multi-stage attacks. Each HMM model has an HMM configuration file associated with it to store its parameters (λ_k)

As discussed previously, each multi-stage attack needs a Markov chain to represent states and stages of the attack. The S parameter in the HMM is composed with these states. Examples of HMM topologies are illustrated in Fig. A.3. The ergodic HMM model, as shown in Fig. A.3b, is an HMM that allows transitions from any state to any other state. Every state in the ergodic model can reach every other state in a single transition. On the other hand, a left-to-right HMM model is a forward-only



(a) Left-to-Right HMM (forward only)

(b) Ergodic HMM (fully connected)

Fig. A.3.: Examples of HMM Topologies

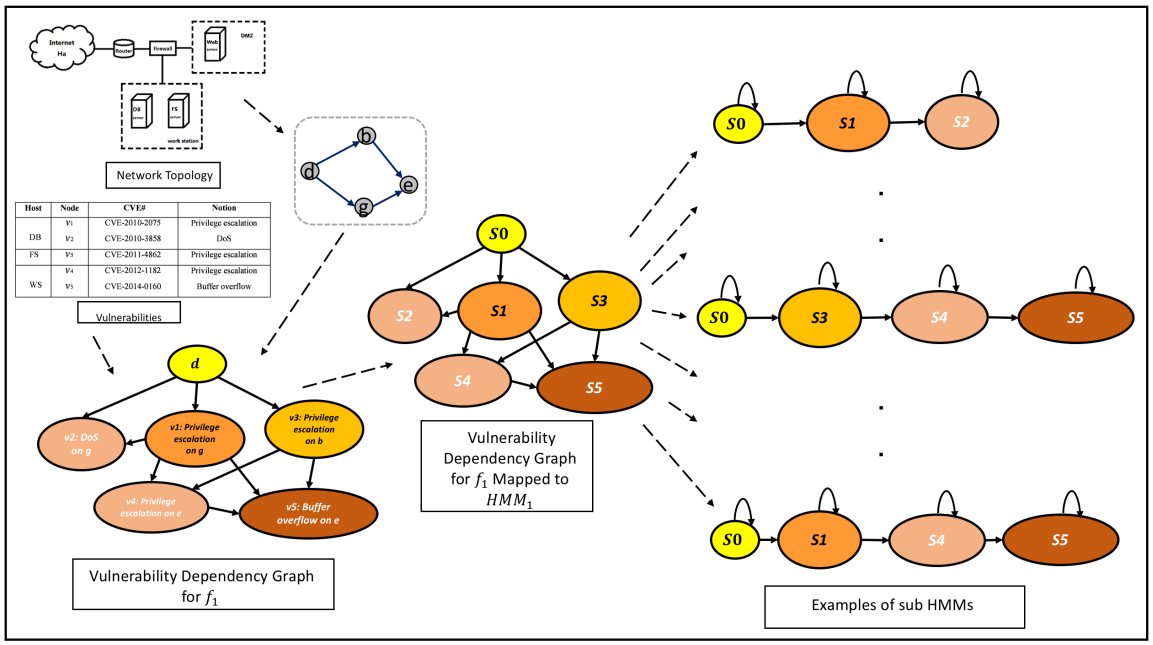


Fig. A.4.: Examples of Multi-stage Attacks - Left-to-Right HMMs

and suitable for modeling order constrained time series whose properties sequentially change over time. and since it has no backward path, the state proceeds from left to right as time proceeds or stays where it was, as shown in Fig. A.3a [52].

The Fig. A.4 shows examples of several multi-stage attacks that can be obtained from the VDG with three and four states and each multi-stage attack has a different HMM model.

VITA

VITA

Tawfeeq A. Shawly is a PhD candidate in the School of Electrical and Computer Engineering at Purdue University. He received his BS and MS degrees in Computer Engineering from the Department of Electrical and Computer Engineering at King Abdulaziz University in Jeddah, Saudi Arabia in 2008. He received his MS degree from the School of Electrical and Computer Engineering at Purdue University in 2016. He is a member of the Center for Education and Research in Information Assurance and Security (CERIAS). His research interests include Security of Networked Cyber-Physical Systems, Game Theory and Machine Learning.