A Framework for Studying New Approaches to Anomaly Detection

Esra N. Yolacan*, David R. Kaeli**

*Eskisehir Osmangazi University, Dept. of Computer Eng.,
**Northeastern University, Engineering College, Dept. of Electrical and Computer Eng.;
e-mail: yolacan@ogu.edu.tr, kaeli@ece.neu.edu

Abstract—In this work, we describe a new framework for an anomaly-based intrusion detection system using system call traces. System calls provide an interface between an application and the operating system's kernel. Since a program frequently requests services via system calls, a trace of these system calls provides a rich profile of program behavior. But we need to use efficient and effective methods while extracting the underlying behavior. In this paper we present an illustrative example to describe how to apply our proposed approach on system call traces for cyber security. We discuss the details of system call anomaly detection by considering various normal behaviors in program traces. Test and detection results show the proposed approach provides fast and accurate anomaly detection by applying context-aware behavior learning.

Keywords—Intrusion detection; anomaly; system call traces.

1. Introduction

The goal of anomaly detection is to identify anomalous behavior, events or items based on deviations from expected normal cases. Anomaly detection is a research area that has been studied extensively for a range of application domains, such as computer and network monitoring for intrusion detection, video and image processing for crowd analytics, activity monitoring for fraud detection, DNA analysis for mutation and disease detection, bio-surveillance for disease outbreak detection, and sensor data analysis for fault diagnosis. Particular anomaly detection processes include outlier detection, novelty detection, deviation detection and exception mining. The processes differ based on the application domain and the employed detection approaches [1]. In our work, we have used the term *anomaly detection* to describe the process of differentiating abnormal behavior from normal behavior in a problem-relevant data set.

Sequential data is a valuable source of information which is available in many aspect of our lives, including weather prediction [2], unusual human action detection in a video [3], pattern discovery [4], [5], and detection of mutations in a gene sequence [6]. Sequences can be discrete or continuous in terms of the value they take for each uniform time interval. A *continuous sequence*, also known as time series, is a sequence of data points which are obtained by measuring a variable at discrete time points [7]. A data point in a continuous sequence may take on any value within a certain range for the measured variable, such as the daily air temperature of a city. A *discrete*

sequence is an ordered series of symbols which can be characters, numbers or words [8]. A data point in a discrete sequence may only take certain values which are limited with an alphabet, such as a gene is a sequence of DNA nucleotides, a program execution trace is the sequence of system calls. In discrete (symbolic) sequences, typically the value of a data point is not meaningful individually, but it provides valuable information when considered with the other symbols in the sequence. In this work, we propose a framework to detect anomalies and demonstrate our approach using a sequence of system call instances as discrete sequences to perform the intrusion detection for cyber security.

We present a general scheme of anomaly detection process in Figure 1. The nature of the anomaly detection process requires a well-defined profile to learn normal behavior. The extracted profile can be anything that can distinguish normal and abnormal behaviors, such as pattern sets, rule sets, probability distributions, statistical models, etc..

A general definition for describing the anomaly detection task on a discrete sequence of symbols can be stated as:

Definition: Given a single sequence (S) as $S = \{s_1, s_2, s_3..., s_n\}$, ($s_i$ is a symbol from a finite alphabet $\sum$), where n is the number of symbols in S and $i = 1, 2, ...n$; then, anomaly detection is the task of deciding whether S is normal or abnormal with respect to learned normal behavior.

Anomaly detection techniques generally use a threshold value to raise an alarm to decide whether

Fig. 1. A general scheme of anomaly detection.

to flag an anomaly. Typically there are two different approaches for the evaluation of anomalies in a discrete sequence of data. The first approach is based on assigning an anomaly score to the entire sequence. If the anomaly score is higher than a predefined threshold, then the sequence is labeled as abnormal [9]. In this approach, normal sequences are expected to have a lower anomaly score than the ones that include an anomaly. An anomaly detection technique in this category needs to apply normalization, compensating for the sequence length to provide a fair evaluation. Although this normalization approach eliminates the impact of False Positives (FP) in a normal sequence, it may also eliminate the True Positives (TP) in an abnormal sequence. For example, if an abnormal sequence is too long, the anomaly score may never reach the threshold value. Therefore, the success of this kind of evaluation depends on the density of abnormal events in the entire sequence and it is difficult to determine where the anomaly starts in a sequence of data.

The second approach is based on performing an evaluation on regions of the sequence to compute an anomaly score [10]. In this approach, abnormal portions of a sequence can be detected when the anomaly score of the evaluated region reaches some predefined threshold. If desired, the anomaly score for the entire sequence can be obtained by combining all anomaly scores assigned to the regions. Three of the advantages of this approach (and which motivate our work) are listed as follow:

- First, since region-based analysis simplifies the data, it allows a wider array of techniques to be applied.
- Second, since this approach works on only a small portion of the data, it enables us to detect local anomalies which would be missed in the first approach.
- Third, since this analysis approach computes

anomaly scores in regions, it performs anomaly detection task without needing to examine the entire sequence.

Typically, an anomaly detection is a time-sensitive task, especially when it is applied for security purposes. When an anomaly is detected, we then need to terminate the anomaly, to eliminate or minimize its effects and investigate the reason that caused the anomaly. Region-based evaluation that works with a partition of sequence is well-suited for real-time anomaly detection applications, but detection still needs to be implemented using efficient algorithms.

In this work, we present an approach that performs a region-based evaluation by using subsequences generated from discrete sequences via a fixed-size windowing technique. We use the term *sequence* to refer to an entire system call sequence of a process in a sequence data set, and the term *subsequence* to refer to a shorter sequence which comprises consecutive system calls.

The paper is structured as follows. Section 2 presents a short review of background and related work in the literature. Section 3 introduces the architecture of proposed intrusion detection system, describes the system call traces, and explains the model training steps. Finally, Section 4 presents conclusions and future work.

2. Background and Related Work

There have been a variety of approaches proposed for anomaly-based intrusion detection using system calls traces. Some of the previous work has focused on using only system call arguments [11], [12], while others have combined the system call sequences with the arguments [13], [14]. But the majority of the previous work in this area has focused on using only system call sequences to train a behavior model. Working only with system call sequences, the various implementations differ in how data is represented. In general, these representations can be grouped into two categories based on their feature extraction methods: 1) frequency-based methods and 2) sequence-based methods.

2.1. Frequency-based methods

Frequency-based feature extraction methods rely on the number of occurrences of each system call. For example, a "bag of words" representation is one such method which is commonly used in text classification. Each trace can be treated as a document and each system call in a document is treated as a word [15]. Since the "bag of words" technique provides a vector-based representation, it is suitable for many machine learning algorithms and widely used in system call anomaly detection [16]–[18]. Instead of only counting the number of occurrences, some approaches improve detection by applying a ranking method based on the relative order of frequency values [19]. Another rich representation of a frequency-based vector is the *term frequency-inverse document frequency* (*tf-idf*), where *term* refers to a system call and *document* refers to a system call sequence [20]. In prior work [21], several forms of *tf-idf* have been considered, applying various classification algorithms on system call sequences and HTPP log data sets.

2.2. Sequence-based methods

Sequence-based methods use the order of system calls or place where a system call occurs in short sequences. A *sliding window* approach is one of the most common techniques used for sequence-based analysis. Forrest et al. [22] extracted normal behavior by sliding a window of size $k+1$ over system call sequences. For each system call, they recorded the following system calls for each position from 1 to k.

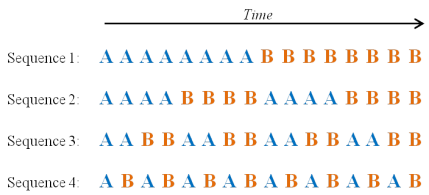

Fig. 2. Example of four different symbolic discrete sequences.

Then, test sequences are scanned and the percentage of mismatches are computed by considering the maximum number of possible pairwise mismatches for a sequence with a lookahead of k. In [23], they improved their prior work by using a "stide" (sequence time delay embedding) system. They first generate a normal database from a set of length k unique short sequences and compute deviations of the test sequence from the normal. Success of both approaches depends on the completeness of the normal data set and the length of the sliding window. One outcome of this prior work was the generation of a benchmark dataset for further system call trace analysis [24].

The effectiveness of an anomaly detection process relies on how well the model is designed. Therefore, the main challenge in anomaly detection is extracting beneficial information from the given sequences. To highlight the importance of the methods selected, we start with an illustrative example. In Figure 2, we have four equal-length discrete sequences generated from a length two alphabet $\sum = \{A, B\}$. Even though it seems obvious that each sequence has different characteristics, it is essential to select an effective method to differentiate between these sequences. In this example, we contrast utilizing vector-based extraction, distance-based and relation-based features to differentiate the sequences.

First, a frequency-based feature vector is computed by counting the number of occurrences of each symbol in the sequence and presented in Table 1. Although the order of symbols in the given sequences is different from each other, the extracted feature vectors are identical, and thus we would be unsuccessful to differentiate between these sequences.

TABLE 1
An Example of Frequency Vector Based Features

Sequences	**A**	**B**
Sequence 1	8	8
Sequence 2	8	8
Sequence 3	8	8
Sequence 4	8	8

Second, the Hamming Distance is used to compute the distances between a sequence and the other sequences. In Table 2, each column shows the distances between the sequence and the other sequences. Each sequence has the same distance to other sequences in the example. While there are more sophisticated and domain specific similarity/distance measures which can be used to evaluate discrete sequences, they do not consider the transitional probabilities between the symbols in a sequence.

Third, relation-based features are extracted by considering the ordering of symbols in a sequence. A fundamental way to perform the behavior extraction process is to calculate transitional probabilities

TABLE 2
An Example of Distance Based Features

Sequences	**Sequence 1**	**Sequence 2**	**Sequence 3**	**Sequence 4**
Sequence 1	0	8	8	8
Sequence 2	8	0	8	8
Sequence 3	8	8	0	8
Sequence 4	8	8	8	0

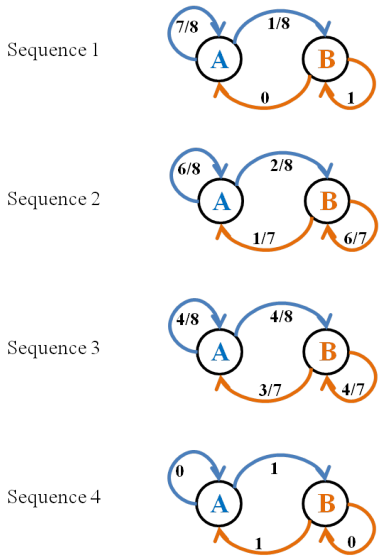

Fig. 3. An example of transitional probability-based features - two-state Markov Chains.

between the symbols in a sequence. We use a first-order Markov process which is the simplest Markov model to represent the transitional probabilities from one state (symbol) to another [25]. In Figure 3, we show the two-state Markov chain used to compute the probability of next symbol, which only depends the current symbol. Using the ordering information present in a symbolic sequence, we will be able to differentiate between sequences and learn the system behavior.

In this work, we applied Hidden Markov Models (HMMs) to generate the normal behavior model of subsequences for intrusion detection. A HMM is a Markov model which can be used when states in a process are not observable, but observed data is dependent on these hidden states. A HMM is well-suited to our problem under these circumstances, because the system call sequences can be considered as observation data which is used to learn and detect the true underlying program behavior.

HMMs have been used many times in the area of anomaly detection-based Host-based Intrusion Detection Systems (HIDS). Hoang et al. [9] decreased the False Positive Rate (FPR) by developing a multi-layer model based on HMMs. The first-layer checks the given test subsequence, and if there is a *mismatch* or if it is a *rare sequence* in the normal data set, then the subsequence is sent to the HMM layer. Yeung and Ding [26] compare a dynamic modeling approach (HMM) with an information-theoretic static modeling approach. They found that the dynamic modeling approaches were more suitable for system call datasets. Du et al. [27] implement a two-state HMM and computes the relative probability of system call sequences to determine if the sequences are normal or abnormal. It has been always an issue to determine the number of hidden states in a HMM. To overcome this issue, Khreich et al. [28] proposed multiple-HMMs (μ-HMMs). They trained multiple models with a varying number of hidden states and combined the results according to the Maximum Realizable ROC (MRROC) method. Although HMMs applied to system call sequences show better results as compared to static approaches, there are still concerns about the required training time. In this regard, Hu et al. [29] proposed a simple data preprocessing approach to speed up HMM training. They improved their previous work [10] with up to a 50% reduction in training time by removing similar subsequences of system calls from the normal dataset.

In our work, we group (clustered) system call sequences by considering the similarities between them to obtain a well trained HMM for each group. This grouping methodology also provides a reduced training time. Moreover, to address the long training time issue in HMMs, we reduced the dataset by selecting only unique system call sequences from the dataset. This reduction also help us avoid identical sequences in training and testing phases of the work.

3. Architecture of IDS

Next, we present our approach to the system call anomaly detection problem. An overview of our framework design is presented in Figure 4. The IDS consists of two phases: 1) training and 2) testing. The implementation of this framework is described in three sections: First, structure of experimentation dataset is detailed. Second, preprocessing and model learning is discussed in the training phase. Third, anomaly detection steps are presented in testing phase.

3.1. System Call Trace Dataset

We evaluate our proposed approach on a well-known system call database provided by the University of New Mexico (UNM) [24]. In UNM benchmark database, each program dataset includes several system call traces which are generated by tracing a number of normal and compromised runs of a program. In this work, we use the UNM and CERT *sendmail* data from the UNM database in our experiments. Each program trace includes system calls associated with the corresponding process IDs (PIDs), since a trace of a program execution

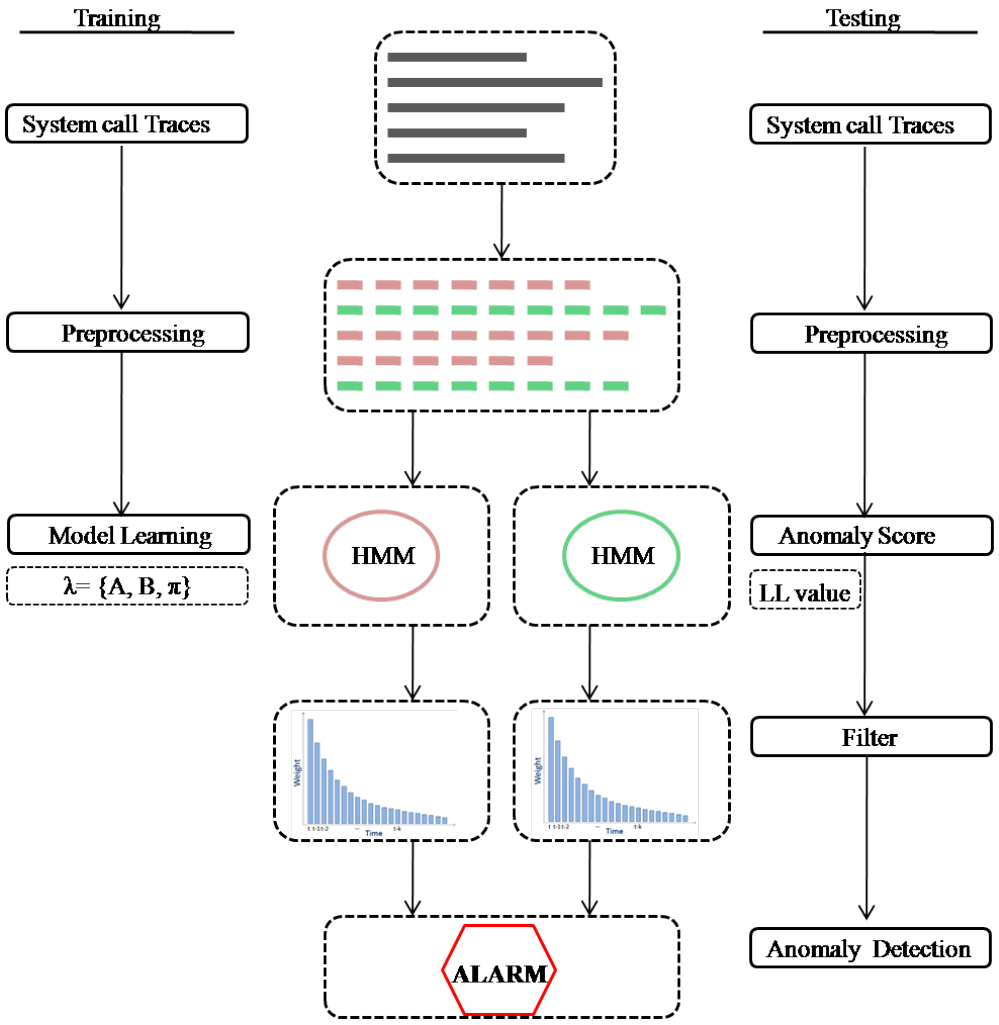

Fig. 4. Design of our system call anomaly detection framework.

TABLE 3
System call sequence for PID:552.

19 105 104 104 106 105 104 104 106 105 104 104 106 5 4 4 5 5 40 40 4 50 5 38 1 105 104 104 106 112 19 19 105 104 104 6 6 106 78 112 105 104 104 106 78 93 101 101 100 102 105 104 104 106 93 88 112 19 128 95 1 5 95 6 6 95 5 5 5 5 5

typically includes multiple processes. In this work, system calls are grouped together according to their PIDs to create an ordered system call list for each process. We applied this PID partitioning on each trace provided in the UNM and CERT *sendmail* datasets. A system call sequence for a process in the UNM *sendmail* dataset is shown in Table 3. In this table, each number represents an index to the system calls matching in the provided mapping file.

While partitioning the program traces according to PIDs of system calls, we store only one copy of a repeated process trace in our dataset. Then, we analyzed unique process traces to detect any structural similarities between processes. This will help us train a HMM for each set of process traces. The concept behind using multiple-HMMs is based on expecting better learning when we have similar training sets. In the rest of this work, training and testing is performed by considering this clustering results. To generate the normal training and test data, we use 10-fold cross-validation for each cluster. Abnormal data is also added to the test sets by considering its cluster.

3.2. Training

In this section we describe model training, whose steps include: 1) *Preprocessing* to differentiate the various contexts in the training dataset and to generate features, and 2) *Model learning* to build a model of normal behavior for each context by training a HMM for each cluster.

3.2..1 Preprocessing

Fixed-length subsequences are generated by using the sliding window technique, as shown in Figure 5. A detailed analysis is needed in order to decide the most appropriate window size to be able to detect anomalies. Researchers have used various window sizes and many of them found that minimum required window size is 6 on *sendmail* program traces [30]–[33]. In other words, a narrower window produces information loss during the subsequence production process. Therefore, features (subsequences) are extracted by using a sliding window length 6 using a step increment of 1. To generate the normal and the test data, we extract unique subsequences for each process trace in the program traces. To perform anomaly detection on a process trace, the subsequences that are extracted from the corresponding process trace are used during the evaluation.

3.2..2 Model Learning

We select to use HMMs for model learning in our anomaly detection framework. A generic HMM structure is presented in Figure 6 which represents the joint probability distribution over states and

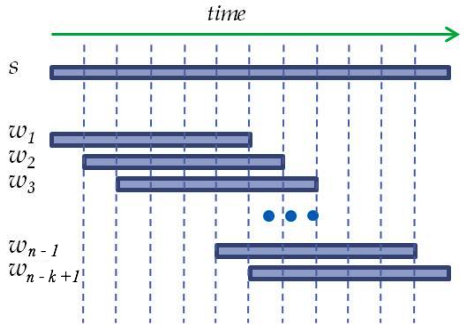

Fig. 5. Subsequence extraction using a sliding window.

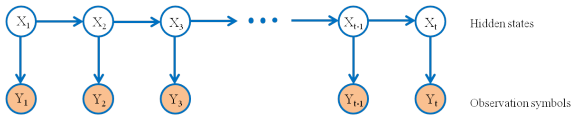

Fig. 6. Hidden Markov Model

observations. HMM is represented in a compact form as $\lambda = \{A, B, \pi\}$, where A is state transition probability distribution, B is observation symbol probability distribution, and π is initial state distribution. There are three kinds of problems one can solve by using HMMs [34].

1) Evaluation: computing the probability of an observation sequence $P[Y|\lambda]$, when the observations sequence Y and a model λ are given.

2) Decoding: estimating the optimal state sequence, when the observations sequence Y and a model λ are given.

3) Training: estimating the model parameters $\lambda = \{A, B, \pi\}$, when the observations sequence Y and the dimensions N and M are given.

In the training phase, we apply the solution of the third problem to find the best-fit model. We define the dimensions and the observation sequences of the HMM to estimate the model parameters $\lambda = \{A, B, \pi\}$. The number of observation symbols is 53, equal to the number of unique system calls in *sendmail* dataset. Subsequences generated via a sliding window are used as observation sequences in the HMM model. We applied Bayesian Information Criterion (BIC) [35] to select the number of hidden states in our model. BIC introduces a penalty term for the number of parameters, while computing a criterion score based on the maximized likelihood. Equation 1 provides the details on how we compute BIC:

$$BIC = -2\,ln(L) + p\,ln(n) \qquad (1)$$

where L is the maximum likelihood, p is the number of free parameters and n is the number of data points. We experimented with the various number of hidden states N. Since it is preferred to use the simplest model that best fits the data, lower BIC scores identify the candidate numbers of the hidden states to be chosen for the model. The lowest BIC values for each fold are found to vary between 40 and 60 hidden states. By considering this range of BIC values, we selected N=53, which is also equal to the number of unique system calls in *UNM sendmail* traces. Then, we trained an HMM for each process set (cluster) using the training subsequences in those clusters.

3.3. Testing

Next, we describe our testing process, whose steps include: 1) *preprocessing* to generate features, 2) computing an *anomaly score* to identify anomalous behavior as deviations from the model of normal behavior, and 3) *anomaly detection* to provide an alarm for the abnormal behavior that deviates from the norm as a possible threat after filtering.

3.3..1 Preprocessing

In preprocessing phase, first we classify the test sequences into one of the previously learned clusters (contexts). Then, we followed our subsequence extraction method on test data as explained in training phase.

3.3..2 Anomaly Score

The parameters of each HMM (one for each context) are estimated in the training phase. We use the relevant HMM for the evaluation, which

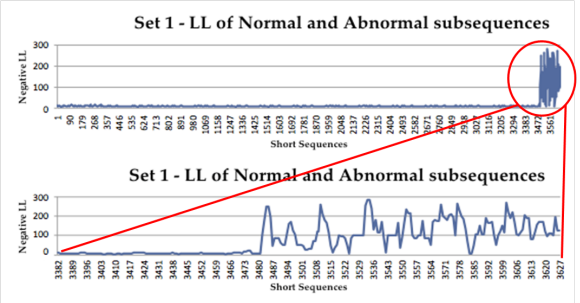

Fig. 7. Anomaly scores (Negative-LL) of Set 1 test subsequences

involves computing the log-likelihood (probability) of an observation sequence. Evaluation is the first problem that one can solve by using HMMs as given in HMM definition. We want to obtain an anomaly score, which means the higher the value of anomaly score, the more likely it is that the abnormal behavior. Since minimizing negative log-likelihood is equivalent to maximizing the log-likelihood ,we compute the negative log-likelihood (LL) values of the observation sequences to obtain an anomaly score for an observation sequence. In Figure 7, we provide an example of anomaly score results for the subsequences of set 1 (1.cluster) normal and abnormal test data.

3.3..3 Anomaly Detection

In the previous step, we obtained an anomaly score for each test subsequence. But, instead of evaluating subsequences individually, an IDS needs to analyze the process trace to decide whether it is normal or abnormal. In order to evaluate a process trace, we applied Exponentially-Weighted Moving Average (EWMA) as a filter on the Negative-LL values of the subsequences within each process. EWMA applies weights on discrete decision values in an exponentially decreasing order to smooth out fluctuations; the most recent values are weighted highest. In our case, decision values are the anomaly scores of subsequences in process traces. To compute EWMA values for each subsequence in a process trace, we used Equation 2:

$$EWMA_t = \alpha\, Y_t + (1-\alpha)\, EWMA_{t-1} \qquad (2)$$

where, Y_t is the decision value (Negative-LL value) at time t, α is the degree of weighting decrease which determines the depth of memory. $EWMA_t$ is the value of the *EWMA* at any time period t.

Figure 8 presents EWMA values for an abnormal process trace. Although the abnormal process behaves normally at the beginning of the trace, it exhibits abnormal behavior after some point. The proposed system call anomaly detection architecture produces alarms if the anomaly score is higher than a threshold value in any point of a process trace. The threshold value is selected through The Receiver Operating Characteristics (ROC) curve, which is found by varying the threshold value on the maximum EWMA filter output of each distinct process trace. ROC plots a curve to show a trade off between false positive rate and true positive rate. The perfect operating point on a ROC curve

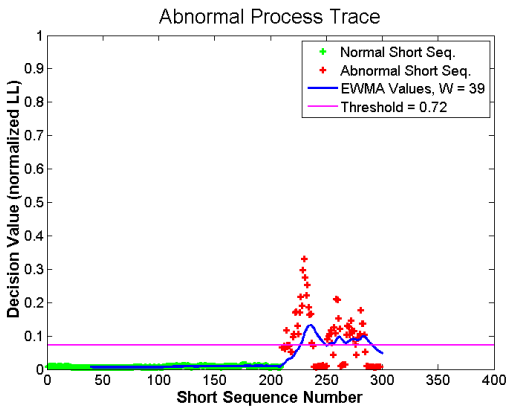

Fig. 8. Time-series plot for an abnormal process trace.

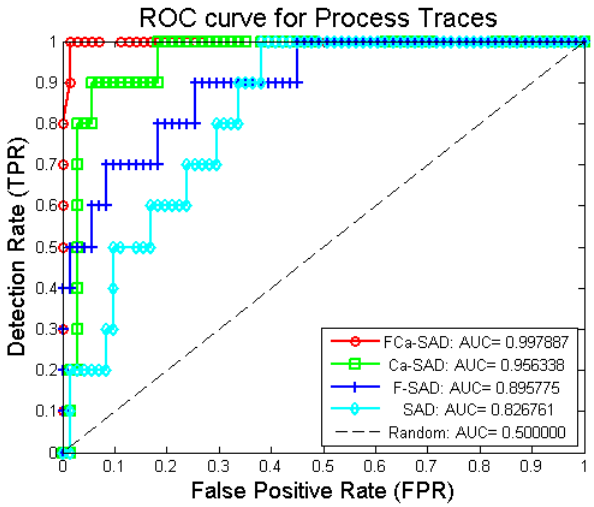

Fig. 9. Process-based evaluation results.

is the (0,1) point where there is 0% false positive and 100% true positive rate. In order to select a threshold, we find the best operating point, which is closest to perfect operating point on the plotted curve.

We examined four different anomaly detection approaches of HMMs on system call traces. SAD (Sequence Anomaly detection) is based on training only one HMM and evaluating the results without EWMA filtering. F-SAD (Filtered-SAD) is based on training only one HMM and evaluating the results with EWMA filtering. Ca-SAD (Context aware-SAD) is based on training multiple HMMs and evaluating the results without EWMA filtering. FCa-SAD (Filtered Context aware-SAD), the proposed method in this work, is based on training multiple HMMs and evaluating the results with EWMA filtering. Figure 9 presents the ROC curves which are found by varying the threshold value on the anomaly score of each distinct process trace. We compute the Area under the curve (AUC) values of each method since AUC is a convenient way of comparing classifiers. Our implementation, FCa-SAD results the highest AUC value when compared with the other three methods.

4. Conclusion and Future Work

In this work, we show how to perform anomaly detection using system call traces with a HMM method. We considered the behavior of system call sequences and explored which preprocessing technique is most suitable for the proposed anomaly detection approach. When compared to conventional techniques, HMM decreases the FPR, since the likelihood value of a normal subsequence is higher than the abnormal ones, even if it does not appear in the normal dataset. Although there has been significant prior work using HMMs for learning program behavior during anomaly detection, in this work we present a new framework for preprocessing traces, leading to better results for anomaly detection. We also provide an on-line anomaly detection scheme that uses a dynamic anomaly score which is computed on each time step using an EWMA filter. This help us to detect the starting point of anomalies in a sequence since the EWMA filter generates an anomaly score at each point of a process trace.

One of the main drawbacks of using HMMs is their significant computation time while training. In order to handle this issue, we reduced the data size by removing repeated sequences, and selected the the shortest possible window length that catches all abnormal subsequences in *UNM sendmail* traces. Our clustering approach also reduces the training time, providing smaller data sizes for each HMM. To differentiate between various behaviors (contexts), we applied similarity-based clustering on system call sequences in the benchmark dataset.

In this work we extend our previous work [36], by providing an architecture to represent our approach and examining a larger data set that includes the CERT *sendmail* data. Directions for future work include investigating other sequential behavior learning approaches for analysis of anomaly, and

examining our approach on a broader set of datasets.

References

[1] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[2] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2011, vol. 526.

[3] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman, "Human focused action localization in video," in *Trends and Topics in Computer Vision*. Springer, 2012, pp. 219–233.

[4] G. Aloysius and D. Binu, "An approach to products placement in supermarkets using prefixspan algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 25, no. 1, pp. 77–87, 2013.

[5] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[6] I. Kinde, J. Wu, N. Papadopoulos, K. W. Kinzler, and B. Vogelstein, "Detection and quantification of rare mutations with massively parallel sequencing," *Proceedings of the National Academy of Sciences*, vol. 108, no. 23, pp. 9530–9535, 2011.

[7] K.-P. Chan and A.-C. Fu, "Efficient time series matching by wavelets," in *Data Engineering, 1999. Proceedings., 15th International Conference on*. IEEE, 1999, pp. 126–133.

[8] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[9] X. D. Hoang, J. Hu, and P. Bertok, "A multi-layer model for anomaly intrusion detection using program sequences of system calls," in *Networks, ICON2003. The 11th IEEE International Conference on*. IEEE, 2003, pp. 531–536.

[10] X. Hoang and J. Hu, "An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 470–474.

[11] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, "On the detection of anomalous system call arguments," in *Computer Security–ESORICS 2003*. Springer, 2003, pp. 326–343.

[12] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, "Anomalous system call detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 61–93, 2006.

[13] F. Maggi, M. Matteucci, and S. Zanero, "Detecting intrusions through system call sequence and argument analysis," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 381–395, 2010.

[14] G. Tandon and P. Chan, "Learning rules from system call arguments and sequences for anomaly detection," in *ICDM Workshop on Data Mining for Computer Security (DMSEC)*, 2003, pp. 20–29.

[15] D.-K. Kang, D. Fuller, and V. Honavar, "Learning classifiers for misuse and anomaly detection using a bag of system calls representation," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 118–125.

[16] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.

[17] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, 2001.

[18] Z. Zhang and H. Shen, "Application of online-training svms for real-time intrusion detection with different considerations," *Computer Communications*, vol. 28, no. 12, pp. 1428–1442, 2005.

[19] S. M. Varghese and K. P. Jacob, "Process profiling using frequencies of system calls," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. IEEE, 2007, pp. 473–479.

[20] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of svm and ann for intrusion detection," *Computers & Operations Research*, vol. 32, no. 10, pp. 2617–2634, 2005.

[21] W. Wang, X. Zhang, and S. Gombault, "Constructing attribute weights from computer audit data for effective intrusion detection," *Journal of Systems and Software*, vol. 82, no. 12, pp. 1974–1981, 2009.

[22] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996, pp. 120–128.

[23] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.

[24] UNM. (2013) Unm system call dataset. [Online; accessed 28-November-2013]. [Online]. Available: \url{http://www.cs.unm.edu/$~$immsec/systemcalls.htm}

[25] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of Markov Chain Monte Carlo*. Taylor & Francis US, 2011.

[26] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[27] Y. Du, H. Wang, and Y. Pang, "A hidden markov models-based anomaly intrusion detection method," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 5. IEEE, 2004, pp. 4348–4351.

[28] W. Khreich, E. Granger, R. Sabourin, and A. Miri, "Combining hidden markov models for improved anomaly detection," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[29] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A simple and efficient

hidden markov model scheme for host-based anomaly intrusion detection," *Network, IEEE*, vol. 23, no. 1, pp. 42–47, 2009.

[30] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 133–145.

[31] E. Eskin, W. Lee, and S. J. Stolfo, "Modeling system calls for intrusion detection with dynamic window sizes," in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, vol. 1. IEEE, 2001, pp. 165–175.

[32] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 2001, pp. 130–143.

[33] K. M. Tan and R. A. Maxion, "Determining the operational limits of an anomaly-based intrusion detector," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 1, pp. 96–110, 2003.

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

[35] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[36] E. N. Yolacan, J. G. Dy, and D. R. Kaeli, "System call anomaly detection using multi-hmms," in *Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on*. IEEE, 2014, pp. 25–30.