Probabilistic Graphical Model on Detecting Insiders: Modeling with SGD-HMM

Ahmed Saaudi, Yan Tong and Csilla Farkas
Department of Computer Science & Engineering, University of South Carolina, 550 Assembly St., Columbia, U.S.A.

Keywords: Insider Threat, Anomaly Detection System, Machine Learning, HMM, Big Data.

Abstract: This paper presents a novel approach to detect malicious behaviors in computer systems. We propose the use of varying granularity levels to represent users' log data: Session-based, Day-based, and Week-based. A user's normal behavior is modeled using a Hidden Markov Model. The model is used to detect any deviation from the normal behavior. We also propose a Sliding Window Technique to identify malicious activity effectively by considering the near history of user activity. We evaluated our results using Receiver Operating Characteristic curves (or ROC curves). Our evaluation shows that the results are superior to existing research by improving the detection ability and reducing the false positive rate. Combining sliding window technique with session-based system gives a fast detection performance.

1 INTRODUCTION

Insiders' misuse of computer systems is a major concern for many organizations. Breach Level Index (Gemalto, 2016), public information of data breaches collected and distributed by Gemalto, asserts that around 40% of data leakage attacks are due to insiders' misuse. The data leakages are scored according to their importance. The risk scores of malicious insider threats are the highest in USA and China: 9.4 and 9.1 respectively. Additionally, the recent studies in (Gavai et al., 2015; House, 2012; Cappelli, 2012; Institute, 2017) show that the insider threat rate has increased compared to 2015. The mean time to detect such malicious data breaches is 50 days (Clearswift, 2018; Cappelli, 2012; Institute, 2017).

There are several solutions proposed to deal with insider threat. Most of them define the suspicious behaviors as low-frequency actions that are performed by a user. So, the unusual behaviors can be compared to high-frequency behaviors to predict the abnormality. The activities can be captured by tracing log data within a specific time unit. The actions' log data can be pre-processed such that it can be modeled using machine learning techniques (Rashid et al., 2016). However, none of these researches address the fact that a long time period is needed to detect malicious behaviors.

In this paper, the raw data from five different domains, "Log on/ Log off," "Connect/ Disconnect," "Http," "Emails," and "Files," are pre-processed to generate new sequence data samples. Multiple domains show different aspects of user behaviors which would support our model to detect malicious behavior. The new data samples are generated according to the detection time unit: Session-based sequences, Day-based sequences, Week-based sequences.

In this paper, we present our results of the session-based analysis.

We propose an unsupervised detection approach to monitor user actions and detect the abnormal behaviors. A user's behavior is represented as a series of activities performed within the organizational environment. To identify the unusual sequence of actions, a stochastic gradient descent version of HMM, "HMM-SGD", is proposed to model the sequence of user activities. The new model has training flexibility because it contains four hyper-parameters. These hyper-parameters can be tuned to improve model convergence.

Our contribution in the presented work can be summarized as:

1. Processing the raw log data to be in session-based, day-based, and week-based sequences. Level granularity data samples help to discover the abnormal behaviours that are distributed over time.
2. Proposing a sliding window technique to consider the effect of the recent history of user activities on their current behavior.

Saaudi, A., Tong, Y. and Farkas, C.
Probabilistic Graphical Model on Detecting Insiders: Modeling with SGD-HMM.
DOI: 10.5220/0007404004610470
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 461-470
ISBN: 978-989-758-359-9

3. Proposing the "HMM-SGD" to model the sequence data samples.

The structure of the remaining sections are as follows:

In section 2, we show related works on insider threat. In section 3, we explain how we implement and train our models to detect insiders. Section 4 provides a brief explanation of the CERT data set. Sections 5 and 6 present the final results of the two models along with the evaluation analysis. Section7 provides a case study similar to the one in (Rashid et al., 2016). Finally, we briefly wrap up our work with the work's limitations and conclusion sections.

2 LITERATURE SURVEY

HMMs have been used with intrusion detection modeling for years. The authors in (Jain and Abouzakhar, 2012) used HMMs to model TCP network data from KDD Cup 1999 dataset and proposed their intrusion detection system. They used Baum-Welch training (BWT) to train the model parameters. To evaluate their model, they applied Forward and Backward algorithms to calculate the likelihood for each sample. Additionally, the Receiver Operating Characteristic curves (or ROC curves) were used to measure the general model effectiveness.Furthermore, the authors in (Lee et al., 2008) proposed a Multi-Stage intrusion detection system using HMM. They evaluated their system by adapting the headmost section data of the "DARPA 2000 intrusion detection" dataset. This dataset provides five different stages or scenarios. They applied HMM on each one of these scenarios independently to create their multi-stage intrusion detection system. The authors in (Rashid et al., 2016), claim to be the first to adapt the Hidden Markov Model to the domain of insiders threats detection. In addition to their application of using the original HMM platform, they proposed a new concept of using a moment of inertia with HMM to improve the results' accuracy. To train and test their work they used the same CERT division dataset as in (Bose et al., 2017), but they used an updated version r4.2. To evaluate their work, they used the ROC curve method. Their highest accuracy using original HMM was 0.797, while their efficiency of using the proposed approach was 0.829.

3 MACHINE LEARNING BASED MODELS

In the presented work, we used sequence-based data samples. Section 4.3 shows how we reformed and generated our data samples or events sequences. We modeled the data samples using the Hidden Markov Model in two different approaches, i.e. the base HMM and HMM-SGD approach.

3.1 Training of Hidden Markov Model

This section illustrates how we train the proposed approach. HMM has three parameters that need to be prepared: initial probability vector (π), transition matrix (A), and emission matrix (B).We use the Baum-Welch algorithm to train the parameters of our model. The Baum-Welch is an HMM context algorithm of the expectation maximization (EM) algorithm. Details of EM algorithm can be found in (Bilmes, 1998). The training process can be set according to the structure of the adapted model. For example, Figure 1 illustrates a four-state structure HMM. We need to find the initial distribution of each of the four states and the transition distribution between them. Also, the distribution of the observed symbols at each state should be determined as well. The list below shows how the model parameters are trained:

1. Initializing model parameters π, A, B with positive random numbers between 0 and 1, where:
   - (π) : The initial distribution of the states. The most probable state that the model will start with.
   - (A) : The initial distribution of the transitions between states.
   - (B) : The initial distribution of the observed symbols.
2. Baum-Welch algorithm is applied to learn HMM parameters. The details of the Baum-Welch algorithm are also presented in (Rabiner, 1989).
3. To make sure that there are no zeros within any of trained HMM parameters, we add a small number to each one of the parameters, followed by a scaling process to ensure the probability condition; all numbers in the symbols matrix add up to one. In addition to that, we use the scaled version of Hidden Markov Model, which also works on overcoming the resolution problems during the training process. Information about the scaled version of HMM is provided in (Rabiner, 1989).

The training process aims to find the model parameters that maximize the likelihood of the sequences that represent the user's normal behavior and

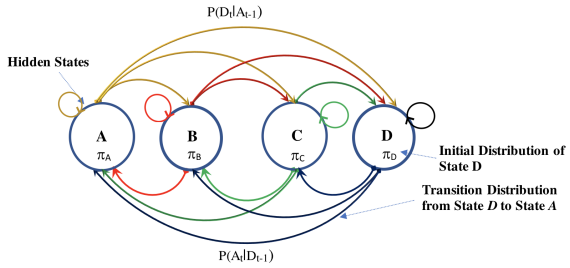


Figure 1: The structure of an insiders threat detection system with HMM.

minimizes the probability of the sequences that represent the anomalous behavior.

3.2 Training with Stochastic Gradient Techniques

As the second approach to model user behavior, we adapt a Hidden Markov Model with Stochastic Gradient Techniques (HMM-SGD) . The main difference in using HMM-SGD is the learning step. In the first method, we use the Baum Welch algorithm to train the model parameters While in this approach we use the stochastic gradient descent (SGD) algorithm to train the HMM. The gradient descent (GD) algorithm is the core algorithm of the training process in the Deep-Learning approaches (the deep neural network) and several others (LeCun et al., 2012). In this approach, we use the SGD method with SoftMax normality function to ensure the probability condition. According to our knowledge, we are the first who use HMM-SGD to solve the insider's threat attack problem.

3.2.1 Selection of GD

The learning methods are divided into two main categories: Stochastic-based and batch-based learning (LeCun et al., 2012). Batch-based learning approaches: needs to process all the training data samples, insider action sequences, to update model parameters. Stochastic based Learning approaches: each single sequence sample is used to update model parameters.

3.2.2 HMM-SGD Learning

Gradient Descent methods are the base of the most successful models, especially in deep learning systems (LeCun et al., 2012). These methods are used to learn parameters during a maximum number of iterations or when there is no change in model performance. The goal of using GD methods is to increase the likelihood of the input data samples, i.e., user activities sequences, given the model parameters. The training procedure works to fit the model parameters with the training dataset such that we can get a high likelihood of a new set of parameters. It is assumed that after scanning all iterations, the parameters will be updated in such a way that the model will converge with a high-objective value.

The objective function is the joint distribution of the hidden state q at the time t and the observed symbols sequences $o_{i1},...,o_{it}$ given the model as described in Equation 1. We train the model to get an objective value for the training sequences. The training samples represent the sequences of user actions within each session as described in section 4.4.

$$
\begin{aligned}
&Objective(sequence_i) = P(Q,O) \\
&= \pi_0(q_0)\prod_{t=1}^{T} P(q_t \mid q_{t-1}) \cdot P(o_t \mid q_t) \\
&\textit{where:} \\
&Q \textit{ is hidden states sequence, } q_t \in \{q_1,...,q_T\}, \\
&O \textit{ is a sequence of the observed symbols,} \\
&o_t \in \{o_1,...,o_T\} \\
&\pi_0 \textit{ is the initial states distribution} \\
&A \textit{ is transition matrix: } A_{i,j} = pr(q_t = i | q_{t-1} = j) \\
&B \textit{ is emission matrix: } B^T_{k=1} = pr(o_t = o_k | q_t = j)
\end{aligned}
\tag{1}
$$

The essential formula of Gradient Descent is illustrated in Equation 2. The GD algorithm uses the chain rule to accomplish the training goal for all model parameters (Theano, 2018). The context of HMM with gradient descent can be summarized as follows:

1. The term W(t) refers to any of the current parameters $\{\pi, B, A\}$.
2. W(t+1) presents the updated version of the model parameters.
3. To orientate the system learning process, we manipulate the learning rate parameter "μ" that changes the learning step during the training procedure.
4. The gradient term of the equation 2 presents the derivation of the objective function with respect to the model parameters.

$$W(t+1) = W(t) - \mu * \frac{\partial Objective}{\partial W} \tag{2}$$

Figure 2 shows the flow diagram of the learning process. First, the model parameters $\{\Pi, A, B\}$ are randomly initialized while maintaining the probability condition, such that all numbers add up to one.

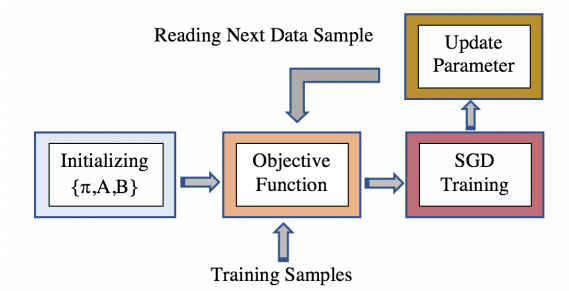


Figure 2: HMM-SGD Learning Process.

The next step is to start modeling the training data samples that involve sequences of the first 50 sessions. The learning procedure is initiated by iterating over a fixed number of iterations. Within each iteration, the objective function will be called to calculate the probability of the session actions sequence as shown in equation 1. The result and the current model parameters will be fed into the gradient descent function. The gradient of the objective function with respect to the model parameters will be calculated. Later on, the parameters will be updated such that we achieve a high probability of the input data sample.

To elaborate more on the SGD training procedure, the training pseudo code is presented in algorithm 1. The main procedure begins by initializing the HMM's parameters. Then, it goes over each of the sequence data samples and calls the training procedure. Algorithm 2 illustrates the training steps that begin by calling the objective function. Then, it updates the model parameters independently by calling the gradient descent function for each of the parameters along with the objective function.

Algorithm 1: Modeling Actions Sequences.

procedure HMM_SGD(inputSequences, hiddenStates, learningRate, iterations)
    Create HMM Object
    Initialize HMM Parameters

        ▷ $\theta_{old} = \{\theta_{\pi\_old}, \theta_{A\_old}, \theta_{B\_old}\}$
    $trainingLength \leftarrow length(inputSequences)$

        ▷ *The first 50 sessions*
    while *iterations* **do**
        for *Training sequences* **do**
            HMM.trainModel(sequence, θ_{model}*, hiddenStates, learningRate)*
                ▷ θ_{model} *is the model current parameters*
        end for
        $iterations - -$
    end while
end procedure

Algorithm 2: Training Procedure.

procedure TRAINMODEL(sample, θ_{model}, hiddenStates, learningRate)

    $Objective = HMM.Objective(sample)$
    $\theta_{\pi\_new} \leftarrow \theta_{\pi\_old} - \mu * SGD(Objective, \theta_{\pi\_old})$
        ▷ θ_{π_old} *is the current initial probability vector*
        ▷ μ *is the learning rate*
        ▷ *SGD is a stocastic gredient descent function*
    $\theta_{newA} \leftarrow \theta_{oldA} - \mu * SGD(Objective, \theta_{oldA})$
        ▷ θ_{oldA} *is the current transition probability matrix*
    $\theta_{newB} \leftarrow \theta_{oldB} - \mu * SGD(Objective, \theta_{oldB})$
        ▷ θ_{oldB} *is the current emission probability matrix*
end procedure

4 DATASET

To test the performance of the proposed approaches, we need a data set that can be used to profile the users' behaviors based on machine log data. For that reason, we used the CERT Insider Threat Data sets (Division and LLC, 2017; Glasser and Lindauer, 2013). The CERT Division cooperated with ExactData, LLC, to create several versions of synthetic insider threat data sets. These data sets are unlabeled sets. They have both synthetic base data and synthetic malicious user data. The data sets project is sponsored by DARPA I2O (Division and LLC, 2017). The CERT data set[1] is a diverse domains data set. It is a public data set that consists of different computer-based log events data files (Rashid et al., 2016). The raw logs data sets include the following computer log events: logon/logoff, the logs of open/closed files, the logs of all surfed websites, the logs of how a user uses the thumb drive Connect/Disconnect, the logs for email messages that have been sent and received, and one file for LDAP information (Rashid et al., 2016; Bose et al., 2017; Division and LLC, 2017). For our work, we used the r4.2 data set that has a huge variety of users event logs. Even though CERT provides r6 data sets, r4.2 has many insider users for several scenarios which is the reason that we adapt r4.2 in this work.

[1] https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099

4.1 The Probability of a Given Sample Sequence

The raw log events are regenerated to be sequences of timed events, as is illustrated in section 4.3. To find the probability of each of the created sequences $Y = (y_1, y_2, ..., y_T)$, we can use one of the two Algorithms: the Naive Based Algorithm, or the Forward-Backward Algorithm.

To use the Naive Bayes method, we need to consider all possibilities of hidden states sequences and add the probabilities across all of them. Using this method is not an efficient way because it increases $O(TN^T)$ runtime. Alternatively, The Forward-Backward algorithm (Rabiner, 1989) is more efficient and it elapses $O(NT^2)$ runtime. In general, the sequence probability is a redundant process of the sum of the product of fraction numbers. The multiplication of two fractions will result in a smaller value. This fact produces small amounts that cannot be processed by computers because of the resolution capability. In many cases, it turns out to be zeros. Thus, in our work, we use the $(-log(P(Y))$ instead of $P(Y)$. The works in (Rashid et al., 2016; Rabiner, 1989) adapt these solutions for the machine resolution problem. In the training section, we will explain two more solutions that we use in our model.

4.2 Malicious Insider's Features

The main purpose that we aim to achieve is to reform the computer-based event log features, from CERT division data sets, which are used with the proposed models. We use two kinds of machine learning models the hidden Markov model and the HMM-SGD. Both of the adopted models work with a sequence-shape data sample. Therefore, we preprocess the multi-domains log events and produce sequences that will be fed to the HMM models. In this paper, we adopt all features that are included with the CERT data set to create session-based, day-based, and week-based data samples of users' action events.

The next section will illustrate the extraction of features and the implementation of the proposed approaches as well.

4.3 Preprocessing of Log Data

At the beginning of this section, we will explain how we preprocess the raw events' log data from CERT datasets. The preprocessing procedure starts from reading the log files from different log domains of each user and ends with the generation of new encoded action event sequences that present user behaviors. The preprocessing phase has four essential stages: Filtration, Encoding, Merging and Extracting. Figure 3 shows the general overview of our preprocessing stages. Each step of the preprocessing is designed to be an independent module. Thus, it can be updated without affecting the other stages.

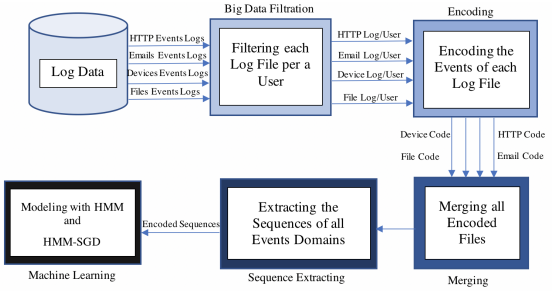


Figure 3: The general platform of the insider threat detection system.

4.4 Features Selection and Extracting

The CERT dataset provides comma separated value (CSV) log files of five different domains (Division and LLC, 2017).

Figure 3 illustrates the main stages of selecting and extracting features. There are four stages (Filtration, Encoding, Merging, and Extracting) that are used to process the raw features. The CERT datasets are big datasets that require a large memory space. For example, a machine with 16 GB RAM cannot hold some of the preprocessing steps especially during the Filtration stage. To overcome this issue, the data filtration performed during loading the CSVs log files from the hard disk.

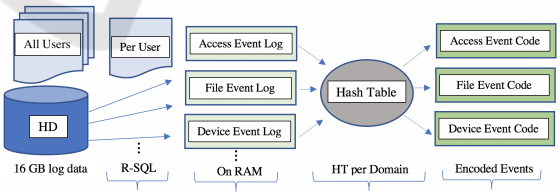


Figure 4: Filtering and encoding multi domains logs data.

The Filtering, Encoding and Merging processes are described as follows:

1. The system starts with filtering the log files of a given user. It uses "R-SQL" to filter the data while loading these files from a hard disk. Using this technique gives the ability to use the available memory size without any issues as shown in Figure 4.
2. The filtered events are encoded sequentially with a hash table . For instance, if the user "AAM0658"

logs in to the system and performs several activities on his machine. The log data of these actions will be encoded as a sequence of numbers. Each number stands for a specific action made by the user.

In HMM context, each code refers to an index in the symbols matrix of Hidden Markov Model. For instance, a representation of one of the encoded session-based symbols of user "AAM0658" are illustrated in Figure 5.

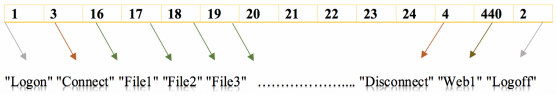


Figure 5: A session sequence of "AAM0658".

3. The encoded events of different domains are merged based on their time-stamp to generate a big vector of symbols as illustrated in Figure 6.

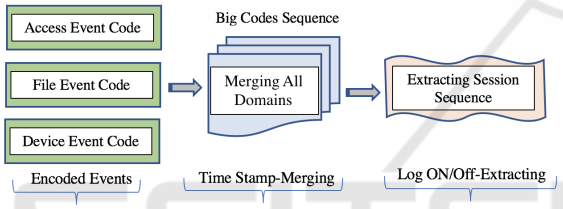


Figure 6: Merging encoded events followed by extracting session-based sequences.

4. The last step is extracting the data samples. The big symbols vector is evaluated to generate three different samples: session-based, day-based and week-based samples. The session-based are determined by tracking the (log on/ log off) events codes. All symbols between log on/log off codes are considered as a session-based sequence.

   The day-based and the week-based samples are obtained by aggregating action events per a day or per a week. Hour, day, week, year and several others features are created to facilitate the pre-processing operations to generate the session-based, day-based , and the week-based sequences. Finally, the resulted sequences are saved in a data frame to be modeled later with HMM and HMM-SGD.

5 MODEL STRUCTURES AND RESULTS

The work presented here utilizes two models: the HMM and the HMM-SGD models. In this section, only the results of the HMM-SGD model is presented because the difference between the performance of the two models is minimal. However, the HMM-SGD model has more tuning flexibility due to the presence of more hyper-parameters.

The structures of HMM and HMM-SGD are also described.

5.1 HMM Structure

HMM experiments were conducted with three hyper-parameters: the number of hidden states, the maximum number of iterations and the number of training samples. The list below shows the combinations of the used hyper-parameters along with the structure of HMMs.

1. HMMs are implemented with $10, 20, 40, 50, 60$ hidden stats.
2. Three detection systems are implemented: the session-based, the day-based and the week-based. Each one of these models is trained using the Baum-Welch algorithm for 20 iterations.
3. We generate the training sets as follows:
   - Session-based system: the first 50 sessions.
   - Day-based: the first 35 day samples.
   - Week-based: the first 5 weeks samples.
4. After the training process, we find the probability of each sequence P(sequence) using the forward algorithm.

5.2 HMM-SGDs' Structure and Results

The HMM-SGD models are trained with four hyper-parameters: the number of hidden states, the maximum number of iterations, the number of training samples, and the value of the learning rate.

The structure of the HHM-SGD models and the hyper-parameter combinations are similar to HMM modes. The only difference is the learning rate. The HMM-SGD models are trained with a 0.01 learning rate.

Similar to the baseline HMM, the probability of each sequence P(sequence) is calculated using the forward algorithm. The results are evaluated using the ROC curve to see the overall performance of the proposed detection approaches.

5.2.1 Session-based Model Results

The session-based sequence is a low level granularity sample. The representation of user actions per session are too narrow to consider many of the users' activities, compared to a day- or week-based sequence.

Moreover, the insiders usually distribute their actions over several sessions so that no one can recognize their anomalous behaviors. However, the session-based system provides the shortest detection time.

It could be observed that the model evaluates most of the labeled sessions with high probability values instead of low probability values.

Although the session-based results look weak, the model shows a very good performance by evaluating the sessions that are near to or surround the labeled sessions with very low probability values, (for more details see study case in section 7). This has inspired us to come up with the idea of a sliding window technique to optimize the model performance.

5.2.2 A Sliding Window Technique

To optimize the models evaluation process, we propose a novel Sliding Window Technique SWT. This approach provides the flexibility to monitor the recent history of user behaviors. For example, to evaluate the session sequence of user "AAM0658" (Figure 5), the sliding window will be used to see the history of the current sequence based on a window size. Thus, instead of considering just the predicted probability of a sequence, a sliding window gives a broad vision of user behaviors.

The proposed technique can also be used to see the future changes of behaviors regarding a current session. For instance, if we want to evaluate session 100 of user "AAM0658", we can see the changes in his behaviors between sessions 100 and 110, using a window of size 10.

In this work, we use the Sliding Window to monitor the recent history of user behaviors for a session-based approach.

6 MODEL EVALUATION

The CERT data set has several scenarios and provides description files for insider's events. These files specify the events that are considered as malicious behaviors. That data is used as truth labels to evaluate the work.

The truth labels of thirty users are used to evaluate the presented work. Those users are insiders according to the definition of scenario one. The insiders attack in scenario one occurs as follows: a User begins to log on after office hours, starts using a removable drive and then begins uploading data to wikileaks.org.

The generated session-based, day-based, and week-based data sets are labeled manually and using a labeling system according to the truth label data.[2]

6.1 Normalization

The baseline of normal behaviors is different among users. We define a baseline as the average of the predicted probabilities of training samples.

To evaluate our work, first, all baselines are normalized to be in the same range. We do that by picking one baseline and shift the rest to be in the same scale. Then, the predicted probabilities of user's behaviors is normalized according the new baseline.

6.2 Evaluation with ROC

The performance is evaluated in term of ROC and Area Under the Carve (AUC). Figure 7 shows the ROC curves of applying sliding window with session-based data samples. It presents four sub figures, each one with different size window. The experiment is implemented with five different model structures and four different window sizes. These include 10, 20, 40, 50, and 60 hidden states and 5, 10, 15, and 20 window sizes. Also, the AUC is presented under each model structure with a different color.

7 CASE STUDY

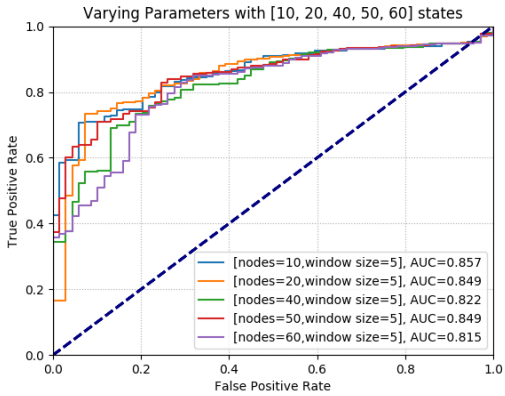
To investigate more about the detection function of the model, a case study is illustrated with more details. The User "MCF0600" is selected as a case study which is the same case study as in (Rashid et al., 2016). As mentioned in section 6, the attack is started when user "MCF0600" begins to log in after office hours, starts using a removable drive, and then begins uploading data to wikileaks.org. User "MCF0600" is considered one of the insiders according to scenario one. The user has malicious behavioral truth labels, so we can use his log data to evaluate the models.

Table 1: User "MCF0600" Data Samples Statistics.

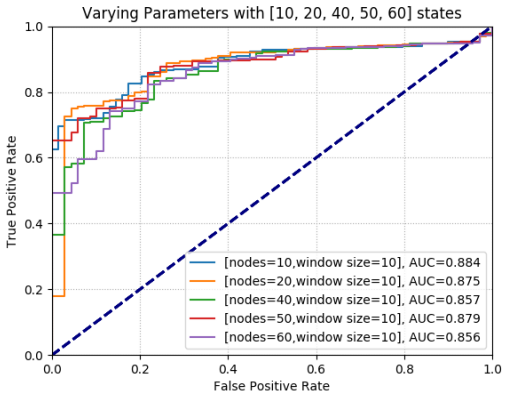
Based	Session	Day	Week
Data samples	308	246	41
Malicious Samples	3	3	1
Training Samples	50	35	5
Malicious Labels	276, 278 , 280	221, 223 , 224	38

Table 1 previews information about the data sample representations for user "MCF0600": session, day

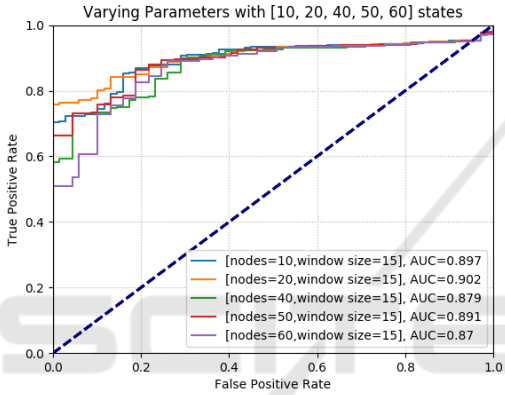
[2]Note: Every procedure in this work is built from scratch. No code is provided from any other work.



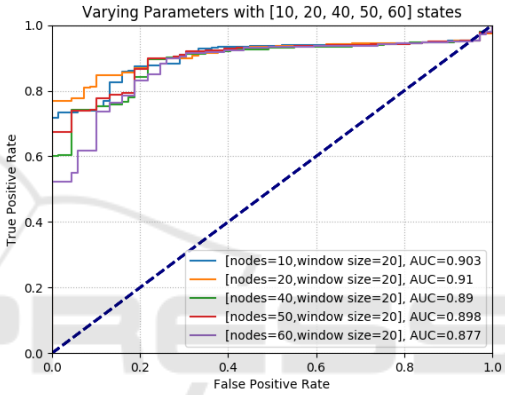
(a): 5 Samples Windows Size.



(b): 10 Samples Windows Size.



(c): 15 Samples Windows Size.



(d): 20 Samples Windows Size.

Figure 7: Session-based HMM-SGD with Sliding Window.

and week-based data samples. The information includes the total number of samples, the number of malicious samples, the number of training samples, and labeled indexes of each data set, section 4.3.

Figure 8 (a), (c), and (e) shows the predicted -Log(Probability) of the three detection systems. The normal samples are colored blue while the malicious samples are colored red. The results are for the testing data sets, So the labeled indexes are different compared to the numbers in table 1.

Figure 8 (b), (d), and (f) presents the histogram distributions of the predicted probabilities with a class color, blue for abnormal behaviors and orange for normal behaviors. The session-based system evaluates the labeled sessions with a high probability[3], sessions 226, 228, and 230. On the other hand, the model evaluates the unusual copying of files or surfing unusual web sites with very low probabilities as shown in figure 8 (a), sessions 221, 232, 235, and 237.

To improve the model performance, SWT is used with the predicted probabilities to analyze the effect of the history behavior on the current behavior. Using SWT shows performance improved compared to the work presented by (Rashid et al., 2016) where each data point represents a week-based behavior. In addition to the data representation, they use the momentum of inertia principle where the detection system is continuously trained based on a predetermined ratio.

In the presented case study, the ROC curve is used with and without a sliding window. The first score is 0.2 while a full score, 1, is the result of using a sliding window. A Window size of 5, 10, 15, and 20 is used and the result is a full score, 1.

Using the sliding window with the session-based system gives a high performance detection system.

[3]High probability means low -Log(Probability).

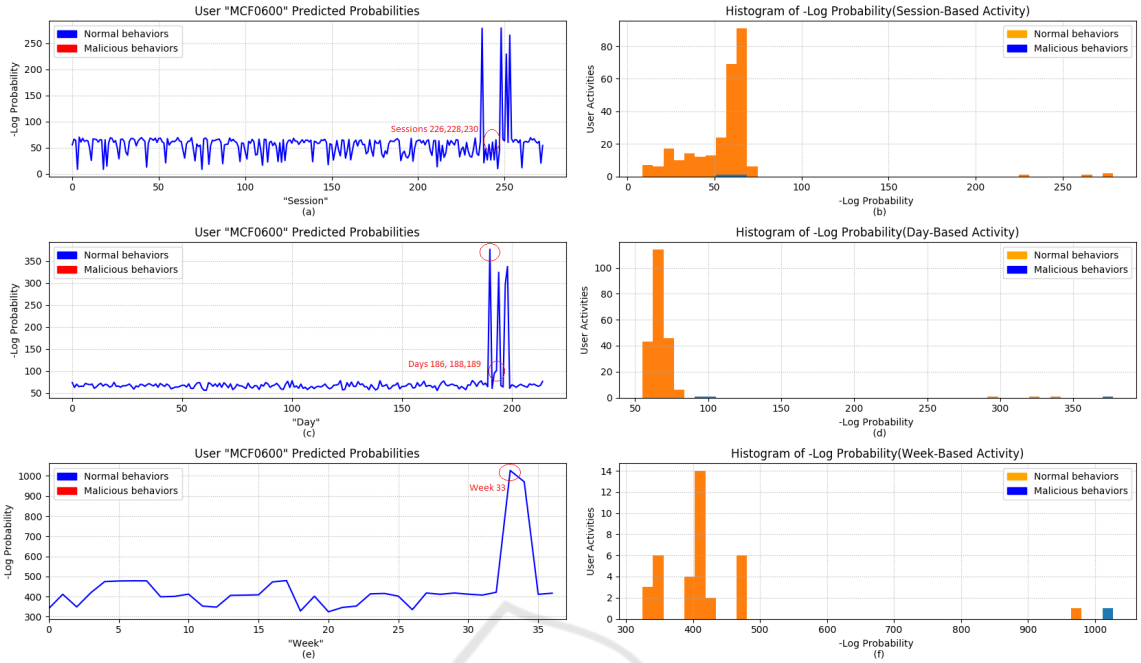


Figure 8: -Log(Probability) and Histograms Plots of "MCF0600" activities with session, day, and week-based data samples.

8 CONCLUSION

This work provides three novel approaches to detect insider threats as described below:

1. The log events are represented in three bases: session-based, day-based, and week-based samples.
2. HMM-SGDs are used to learn users' normal behaviors. The learned models work as a baseline to investigate the behavior of new samples.
3. A novel sliding window technique is proposed to monitor the history of a user behaviors and detect malicious insiders effectively.

It was concluded that combining the session-based approach with a sliding window technique provides a better detection capability compared to existing works.

REFERENCES

Bilmes, J. (1998). A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *Retrieved January 18, 2008, from the International Computer Science Institute Web site: http://ssli.ee.washington.edu/people/bilmes/mypapers/em.pdf*, 1198(510).

Bose, B., Avasarala, B., Tirthapura, S., Chung, Y. Y., and Steiner, D. (2017). Detecting Insider Threats Using RADISH: A System for Real-Time Anomaly Detection in Heterogeneous Data Streams. *IEEE Systems Journal*, 11(2):471–482.

Cappelli, D. (2012). *The CERT Guide to Insider Threats*. Pearson Education,Inc.

Clearswift (2018). insider threat 74 security incidents come extended enterprise not hacking groups.

Division, C. and LLC, E. (2017). Insider threat tools.

Gavai, G., Sricharan, K., Gunning, D., Rolleston, R., Hanley, J., and Singhal, M. (2015). Detecting Insider Threat from Enterprise Social and Online Activity Data. *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats - MIST '15*, pages 13–20.

Gemalto (2016). Breach level index data breach database and risk assessment calculator.

Glasser, J. and Lindauer, B. (2013). Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. *2013 IEEE Security and Privacy Workshops*, pages 98–104.

House, T. W. (2012). Presidential Memorandum – National Insider Threat Policy and Minimum Standards for Executive Branch Insider Threat Programs.

Institute, P. (2017). Cost of Cyber Crime Study.

Jain, R. and Abouzakhar, N. S. (2012). Hidden markov model based anomaly intrusion detection. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 528–533.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K. R. (2012). Efficient backprop. *Lecture Notes in Com-*

puter Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7700 LECTU:9–48.

Lee, D.-h., Kim, D.-y., and Jung, J.-i. (2008). Multi-Stage Intrusion Detection System Using Hidden Markov Model Algorithm. *2008 International Conference on Information Science and Security (ICISS 2008)*, pages 72–77.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition.

Rashid, T., Agrafiotis, I., and Nurse, J. R. (2016). A New Take on Detecting Insider Threats : Exploring the use of Hidden Markov Models. *MIST '16 Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, pages 47–56.

Theano (2018). Gradient kernel description. http://deeplearning.net/software/theano/extending/op.html?highlight=grad#grad. Accessed: 2018-02-12.