IEICE TRANS. INF. & SYST., VOL.E103–D, NO.4 APRIL 2020

LETTER

Against Insider Threats with Hybrid Anomaly Detection with Local-Feature Autoencoder and Global Statistics (LAGS)

Minhae JANG[†a)], **Yeonseung RYU**[††b)], **Jik-Soo KIM**[††c)], *Nonmembers*, *and* **Minkyoung CHO**[††d)], ***Member***

SUMMARY Internal user threats such as information leakage or system destruction can cause significant damage to the organization, however it is very difficult to prevent or detect this attack in advance. In this paper, we propose an anomaly-based insider threat detection method with local features and global statistics over the assumption that a user shows different patterns from regular behaviors during harmful actions. We experimentally show that our detection mechanism can achieve superior performance compared to the state of the art approaches for CMU CERT dataset.

key words: *abnormal detection, sequence-to-sequence learning, autoencoder, reconstruction error*

1. Introduction

Insider threats caused by stakeholders who have access to internal systems, can severely damage the organizations with leakage or destruction of sensitive or commercially valuable information/systems. Unlike other cyber attacks, it is very difficult to detect these threats because an internal attacker has permissions to access the information/systems, and does not need to bypass security devices such as IDS/IPS, or Firewalls. Over the last two decades, insider threat detection has been heavily researched [1], and "anomaly-based detection method" is one of the popular mechanisms in this area. The underlying assumption is that a malicious insider behaves differently from his/her normal patterns when they try to thieve or attack internal information/systems (e.g., collecting a large amount of information, uploading files to outside website, or periodically visiting websites for job search).

In this paper, we propose an insider threat detection method based on a sequence-to-sequence *autoencoder* model. Autoencoder [2] basically learns to make output as same as the input itself and it is one of the unsupervised learning mechanisms. Therefore, autoencoder does not need to label each data. In order to detect insider threats, sufficient amount of data must be obtained along with the case definitions of insider threats. However, insider threat cases may change widely according to varying time and circumstances, so that it is almost impossible to accumulate enough data to cover all these cases. Instead, we try to learn normal behaviors from data, and consider the potential candidate of insider threats based on the deviation from trained normal behaviors.

User behavior data is event-based time series data, and the data length is different from each other depending on user and time. To learn this kind of data, we use *a sequence to sequence model* [3] based on RNN (recurrent neural network). Although RNN-based model shows excellent performance for sequence data, it is already known that the performance gets degraded when the sequence length becomes very long. Therefore, instead of learning the whole sequence, we divide it into relatively short fixed-length sequences, and learn *local* sub-sequences with a RNN model. One minor problem in this approach is that these local subsequences may not represent the main characteristics of the whole sequence. To address this problem, we have leveraged additional statistical techniques (e.g., standard deviation factors) to effectively capture the *global* characteristics for the entire sequence.

2. Related Work

This section mainly focuses on the related studies of insider threat detection using CMU CERT insider threat test dataset [4].

Rashid et. al. [5] proposed a novel method to detect insider threats by using Hidden Markov Model (HMM). Given the user computer usage logs which contains detailed information about login / logoff, web access, USB connection, and email, they transform those logs to simple integer sequences by categorizing to seven event types. Then, they trained the hidden markov model for the first 5 weeks which are considered as a user's normal behavior. They detect the suspicious threats by estimating if events are significant deviations from the normal behavior. In their experiments, the performance of their algorithm is less than 0.8 in the aspect of AUC (Area Under Receiver Operating Characteristics Curve).

Ha et. al. [6] proposed a deep learning based approach for insider threats detection. They followed the same preprocessing method as Rashid et. al. and learning the sequences with RNN autoencoder model, inspired by Encoder-Decoder-Anomaly Detection (EncDec-AD) [7]; this detects an anomaly pattern based on autoencoder's re-

Manuscript received September 30, 2019.
Manuscript revised November 20, 2019.
Manuscript publicized January 10, 2020.

[†]The author is with Digital Solution Lab., KEPCO Research Institute, Korea.
[††]The authors are with Dept. of Computer Engineering, Myongji University, Korea.
a) E-mail: minhae.jang@kepco.co.kr
b) E-mail: ysryu@mju.ac.kr
c) E-mail: jiksoo@mju.ac.kr
d) E-mail: mkcho@mju.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2019EDL8180

construction errors. They split a whole sequence into small fixed-size sequences, and then learn them with RNN autoencoder. Evaluation results show that the proposed model clearly outperforms HMM-based models. For additional performance improvements, they used the frequency of USB devices as an extra information.

Tuor et. al. [8] proposed an insider threat detection method with stacked LSTM models. Instead of simply preprocessing seven event types, they analyze more fine-grained events consisting of a total 408 detailed types according to usage time, existing email attachment, file operations (e.g., reading, writing, copying, and deletion), etc. In the experiments with CMU CERT r6.2 dataset, they showed that their model outperformed SVMs and PCA which are the standard anomaly detection techniques. Also, they argue that 90% of insider threats can be detected if they examined 250 instances with top anomaly scores per day. Another recent work by Yuan et. al. [9] enumerated events into 16 types and tested with various LSTM models and CNN models. One of a LSTM with CNN model successfully detected insider threats and obtained AUC = 0.9449 in the best case at CMU CERT r4.2 dataset.

This paper extends our previous work [6] by effectively fine-tuning the model with attention mechanism, and revising the seq2seq learning model. In addition, we have eliminated the application-specific information such as USB usage frequency for generalization. Instead, we introduce standard deviation factors to estimate each event's global characteristics which can be easily extended to support general-purpose sequence data sets. Our extensive experimental results show that proposed insider threat detection scheme can clearly outperform existing approaches for CMU CERT dataset.

3. Architecture and Implementation

In this section, we describe our target datasets, preprocessing mechanism, proposed learning models, and anomaly measurements.

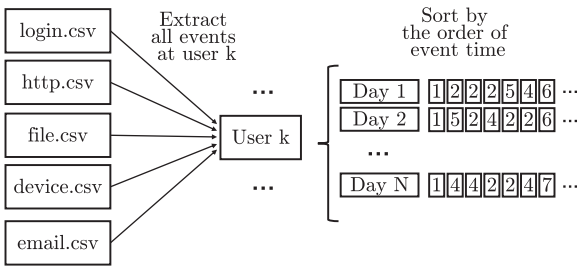
3.1 Data Sets

CMU CERT dataset contains detailed logs for user's events. Each event has transaction ID, timestamp, user ID, system ID, and other event specific details (as seen from Fig. 1).

To handle these event data, we preprocess them as depicted in Fig. 2. The input log data consist of 5 event files related to login, http, file, device, and email. We extract events by user ID from all files and merge them into a single file for each user. After sorting the user file in a chronological order, the final event sequence data can be created through enumeration process; Given the input data, we simply enumerate events into seven types, and detailed information are ignored (as we can see from Fig. 2). The enumerated values of events start from 1, and 0 is used as a padding value indicating that there is no event data.

```
logon,{U1Q4−I9PX02PE−8712XJDX},02/24/2011 19:49:41,
      WDD0366,PC−0155,Logon
device,{Q613−A2AY45RM−4749NTDA},02/24/2011 22:07:28,
      WDD0366,PC−0155,Connect
http ,{S3I7−R3CN14RD−1922CYQD},02/24/2011 22:29:04,
      WDD0366,PC−0155,http://wikileaks.org/Julian_Assange
device,{J3N5−J1IG44ZN−4509ZXOG},02/24/2011 22:31:31,
      WDD0366,PC−0155,Disconnect
email,{I3G5−W1FL72KE−1363JNAZ},08/31/2010 11:21:01,
      XHW0498,PC−9840,Selma−Burch@harris.com,,,Xerxes.
      Howard.Wilcox@dtaa.com,28488,1
file ,{Y8V1−M0YK81DK−8272ZWUF},11/04/2010 15:19:09,
      MPM0220,PC−2344,AZTCQ3OG.exe,4D−5A−90 file
      keylogging malware
logon,{V5E4−F1JY53FG−8030MANZ},02/24/2011 23:11:08,
      WDD0366,PC−0155,Logoff
```

Fig. 1 Examples on input logs



[Event type enumeration]

Logon	Http	Email	File	Conn.	DisCon.	Logoff
1	2	3	4	5	6	7

Fig. 2 Preprocessing of CMU CERT Data

3.2 Seq2seq Autoencoder Model with Attention

In order to deal with user's event-series data, we have built an autoencoder based on "sequence-to-sequence (seq2seq) model". An autoencoder basically consists of an *encoder* part that maps an input into a latent space typically smaller than input dimension, and a *decoder* part that restores the latent space to its original data form. After training, the autoencoder has a high recovery rate for the patterns learned, and may fail to reconstruct for non-trained patterns.

seq2seq model is to learn how to transform one input sequence into another one. There are many real world applications for this type of seq2seq learning such as language translation and word recommendation. In our case, since we are making input and output sequences the *same*, the seq2seq model operates as autoencoder. Specifically, we designed our seq2seq autoencoder by taking the encoder's final state (which is a kind of summary of an input sequence) into the initial state of decoder. Also, in order to improve the overall performance, additional *attention* mechanism is effectively applied to our model. Our seq2seq model with attention can utilize all state information of each input sequence along with encoder's final state (as we can see from Fig. 3).

3.3 Anomaly Detection on Long Sequence

Let S denotes a set of sequences where each sequence s_i

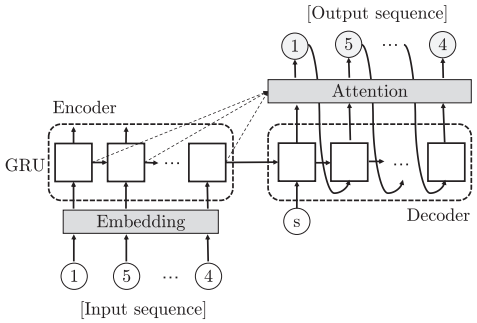


Fig. 3 Sequence to sequence model with attentions

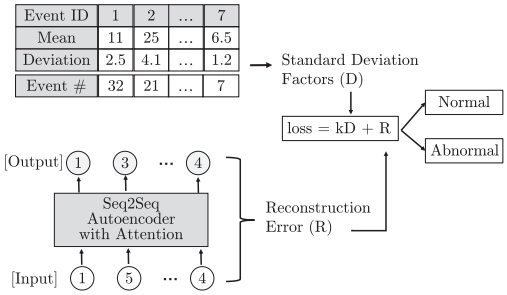


Fig. 4 Abnormal detection with s.t.d factors and reconstruction errors

Table 1 CERT r4.2. data

total user	1,000 users
total days	500 days (2010-01-02 - 2011-05-17)
total events	32,770,227 events
threat events	7,323 events
threat users	70 users
threat (user, day)	986 pairs

consists of a series of a discrete element e. Let n denote the number of the sequences and len$(\cdot)$ denotes a function to compute the length of a given sequence.

$$S = \{s_1, s_2, \cdots, s_n\} \text{ where } s_i = [e_1, e_2, \cdots, e_{\text{len}(s_i)}]$$

Let us consider the application of our cases where the length of sequence s_i is long enough (e.g., over 200 events for each day) and the number of sequences, n, is relatively small (e.g. 60 days). Then, it is hard to train the sequence with our seq2seq autoencoder due to the number of training data is too small. In order to train this kind of data, we inflates the given data set by extracting subsets of a sequence through sliding with a small sized window. Let T_{s_i} denotes a set of subsequences generated from a sequence s_i where w is the size of sliding window. Then, we can effectively augment the sequence data s_i as followings:

$$T_{s_i} = \{[e_1, e_2, \cdots, e_w], [e_2, e_3, \cdots, e_{w+1}], \cdots\}$$
$$\text{where } e_j \text{ is the } j\text{-th element in sequence } s_i$$

Now, given a set of sequence S, we define a loss function, called *reconstruction error* R, of our seq2seq autoencoder. R can be compute as the sum of categorical crossentropy on S as follows.

$$R = \sum_{s_i \in S} \sum_{t_j \in T_{s_i}} \sum_{e_k \in t_j} e_k \log \Pr(e_k)$$

However, this reconstruction error may miss the characteristics of an entire sequence because the training is performed only with subsequences, not an entire sequence. In order to reflect the overall sequence characteristics, we introduce a *standard deviation factor*; how far is the number of occurrences of each event from the given mean and standard deviation of normal sequences. We consider it as a loss when the factor exceeds some threshold τ as followings:

$$D = (X - m)/\sigma \quad \text{if } (X - m)/\sigma \geq \tau$$
$$= 0 \quad \text{otherwise}$$

where m, σ are mean and standard deviation, respectively. In our experiments, the threshold τ value is set to 2 which is typically about twice the standard deviation (a half-width of the 95% confidence interval). Within this confidence interval, it is considered as a normal behavior with no penalty (i.e., $D = 0$)

Finally, the loss function can be defined with reconstruction error R from our seq2seq model and the standard deviation factors D_e of each element e

$$\text{loss} = R + k \cdot \sum_{e \in E} D_e \text{ where } E \text{ is a set of event types}$$

where k is a hyper-parameter constant to balance the reconstruction error and the standard deviation factors. If the loss is bigger than some threshold given by user, we consider the example as *abnormal* indicating a candidate of potential insider threats.

3.4 Implementation

We implemented our model with Keras in Tensorflow 1.13 and performed experiments on Ubuntu 16.04 desktop 64bit OS with Intel i7-8700 (12 Core CPU), 64GB Memory and NVIDIA GTX Titan Z (vram 12GB) GPU. The hyper-parameters for optimizations in our model have been selected as the learning rate to 0.05, the nesterov option to be true, and the momentum to 0.9. For the details of our implementation, refer to the source code repository [10].

4. Evaluation

Our main dataset is CMU CERT revision 4.2 which contains relatively a lot of abnormal events compared to other revisions. Basically, a thousand of users generated about 32 million computer usage events during 500 days. The total number of threat events is 7,323 and the distinct threat pair by user and day is 986. Our objective is to find 986 (user, day) threat pairs among the total 1,000 users $\times$ 500 days pairs on the dataset (actually 330,452 pairs).

For each user, we built and trained a seq2seq autoencoder model. The training data is the first 60 days of user behavior logs under the assumption that users act normally during this period. The maximum epoch is 500 and the training can be stopped early if the loss becomes below 0.002 in

Table 2 The confusion matrix for our model

	precision	recall		$\hat{T}$	$\hat{F}$
ours	54.02%	85.80%	T	846	140
(size=3)	99.96%	99.78%	F	720	328,746
ours	41.78%	88.95%	T	877	109
(size=4)	99.97%	99.63%	F	1,222	328,244
ours	37.73%	89.15%	T	879	107
(size=8)	99.97%	99.70%	F	1,451	328,015
ours	14.20%	91.38%	T	901	85
(size=16)	99.97%	98.35%	F	5,446	324,020

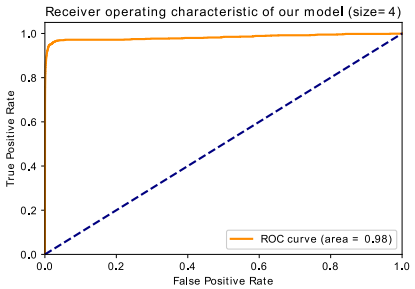


Fig. 5 ROC curve for our model (size = 4)

order to prevent overfitting. As performance metrics, we employ a confusion matrix, ROC (Receiver operating characteristic) curve, and CR-k [8]. CR-k measures the maximum number of checking to detect all instances of threats.

First, to investigate the effects of subsequence sampling, we test our model with various training sequence lengths by changing the sizes of sliding window. The experimental results are presented in Table 2. Note that T means a threat event and F denotes a normal event. $\hat{T}$ and $\hat{F}$ show the events that are detected as threats and normal activities respectively.

Our goal is to maximize the *true positive* (T, $\hat{T}$) and minimize the *false positive* (F, $\hat{T}$). In Table 2, we can observe that as the pattern size becomes larger, the true positive (T, $\hat{T}$) increases, however the false positive (F, $\hat{T}$) also rapidly increases. As the pattern size is increased, the possible patterns will be increased exponentially (e.g., 7^3, 7^4, 7^8, 7^{16}). Therefore, it may be difficult to learn user patterns, not quite strongly depending on near events. Although the precision and recall seem to be too low, due to the huge number of total (user, day) pairs (i.e. 328,669), our performance can be considered as relatively good enough. Based on these results, we choose the best sliding window size as 4, which can show high true positive and relatively reasonable false positive rates.

As we can see from Fig. 5, our proposed model shows a good shape in the ROC curve. In addition, we could achieve AUC value of 0.9855 which is better than the recent research results in Yuan et. al. [9] where they used a stacked RNN and CNN resulting in 0.9449 AUC value in the best case.

To compare with another recent work by Tuor et. al. [8] based on the CR-k metric, we test our model with CMU CERT r6.2 by following their experimental settings. They chose CR-250 where the maximum number of checking to detect all threats is 250. On the other hand, our model needs

Table 3 The confusion matrix at CMU CERT r6.2

	precision	recall		$\hat{T}$	$\hat{F}$
ours	1.05%	90.00%	T	18	2
(size=4)	99.29%	99.99%	F	1,714	242,032

at most 40 detections per day, that is CR-40, which is six times less than the previous work's CR-250.

Note that previous papers did not specify their accuracy results so that we cannot directly compare them with ours.

5. Conclusion

In this paper, we have addressed the insider threat detection problem with abnormal detection deep learning model. We trained local feature autoencoder based on sequence to sequence model with attention and used standard deviation factors to capture global statistical characteristics. Experimental results show that our proposed model can outperform the previous mechanisms and achieve the best available results, especially at CMU CERT dataset. Our approach is not limited to CMU CERT dataset only, and can be easily extended to various sequence dataset with discrete elements such as temperature, stocks, heartbeats, etc.

Acknowledgments

This research was supported by 2018 Research Fund of Myongji University, and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1005360).

References

[1] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: A survey," IEEE Commun. Surv. Tutorials, vol.20, no.2, pp.1397–1417, 2018.
[2] G.E. Hinton and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol.313, no.5786, pp.504–507, 2006.
[3] I. Sutskever, O. Vinyals, and Q.V. Le, "Sequence to sequence learning with neural networks," NIPS, pp.3104–3112, 2014.
[4] J. Glasser and B. Lindauer, "Bridging the Gap: A pragmatic approach to generating insider threat data," 2013 IEEE Security Privacy Workshops, pp.98–104, IEEE 2013.
[5] T. Rashid, I. Agrafiotis, and J.R.C. Nurse, "A new take on detecting insider threats: Exploring the use of hidden markov models," Proc. 8th ACM CCS International Workshop on Managing Insider Security Threats, pp.47–56, ACM, 2016.
[6] D.W. Ha, K.T. Kang, and Y. Ryu, "Detecting insider threat based on machine learning: Anomaly detection using rnn autoencoder," J. KIISC, no.4, pp.763–773, Aug. 2017.
[7] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," arXiv preprint arXiv:1607.00148, 2016.
[8] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," Proc. AI for Cyber Security Workshop at AAAI 2017, 2017.
[9] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," International Conference on Computational Science, vol.10860, pp.43–54, Springer, 2018.
[10] "LAGS," https://github.com/minkcho/lags.