Insider Threat Detection Using Multi-autoencoder Filtering and Unsupervised Learning

Yichen Wei, Kam-Pui Chow, Siu-Ming Yiu

Chapter 15

INSIDER THREAT DETECTION USING MULTI-AUTOENCODER FILTERING AND UNSUPERVISED LEARNING

Yichen Wei, Kam-Pui Chow and Siu-Ming Yiu

Abstract Insider threat detection and investigation are major challenges in digital forensics. Unlike external attackers, insiders have privileges to access resources in their organizations and violations of normal behavior are difficult to detect.

This chapter describes an unsupervised deep learning framework for detecting insider threats by analyzing system log files. A typical deep neural network can capture normal behavior patterns, but not insider threat behavior patterns because of the presence of small, if any, amounts of insider threat data. For example, the autoencoder unsupervised deep learning model, which is widely used for anomaly detection, requires a dataset containing labeled normal data for training purposes and does not work well when the training dataset contains anomalies. In contrast, the framework proposed in this chapter leverages unsupervised multi-autoencoder filtering to remove anomalies from a training dataset and uses the resulting trained Gaussian mixture model to estimate the distributions of encoded and recognized normal data; data with lower probabilities is identified as insider threat data by the trained model. Experiments demonstrate that the multi-autoencoder-filtered unsupervised learning framework has superior detection performance compared with state-of-the-art baseline models.

Keywords: Insider threat detection, unsupervised deep learning, autoencoders

1. Introduction

The insider threat continues to cause significant losses to governments, businesses, hospitals and educational institutions. Insiders are masqueraders, traitors [22] or unintentional violators whose behaviors are abnormal compared with their organizations' computer system conventions.

Malicious insiders are difficult to detect because they are located within their organizations and have privileges to access resources in their organizations.

Insider threat detection and investigation are challenging tasks in digital forensics. Malicious insiders may take actions such as inserting backdoors in internal systems to launch attacks later, installing keyloggers to gain credentials and steal sensitive information, even deleting traces of their unauthorized activities. According to the Breach Level Index [28], almost 214 sensitive data records are compromised every second in the world and around 40 percent of the compromises are due to insiders [20].

Digital forensics is *a posteriori* in nature – investigations are conducted after crimes were committed and the damage has been done. To address the insider threat, it is necessary to make *a priori* predictions with the help of deep learning methods that automatically detect anomalous user behavior and capture evidence of malicious activity.

This chapter presents a novel unsupervised deep learning insider threat investigation framework that can profile normal user behavior patterns and prevent data leakage. A synthetic insider threat dataset from the Software Engineering Institute at Carnegie Mellon University [26] is employed to evaluate the insider threat detection framework. The original log files in the dataset are pre-processed to extract daily system operation features and user metadata [29] that are used to distinguish insider threat activities from normal activities. Unfortunately, the dataset contains very limited, if any, insider threat data, which makes it difficult for traditional supervised deep learning models to learn insider threat behavior patterns.

The proposed framework employs a neoteric unsupervised deep learning model that is inspired by the basic autoencoder model [10]. The framework leverages unsupervised learning to solve the detection problem. It is based on the intuition that an autoencoder may not learn feature patterns well if it seldom or never observes insider threat behavior patterns; in other words, the reconstruction error of insider threat data would be large. Therefore, multiple autoencoders are cascaded to filter out data with large reconstruction errors as potential insider threat data, leaving the dataset with normal data. Following this, a Gaussian mixture model is employed to estimate the distribution of the recognized encoded normal data.

Experiments demonstrate that the proposed framework compares favorably with state-of-the-art unsupervised insider detection methods. Specifically, the framework increases the recall and area under the ROC curve (AUC) metrics by more than 19% and 23%, respectively.

2. Related Work

Insider threat detection has been studied widely by academia and industry. The Software Engineering Institute at Carnegie Mellon University has done considerable work on detecting insider threats. For example, researchers have inspected network traffic through the Squid proxy server [25], set up access control lists and signatures, and tagged documents to check if data leakage has occurred from within an organization. Splunk [27] has created mature security products for log management and anomaly detection.

In general, there are two broad insider threat detection solutions: (i) traffic inspection; and (ii) behavior profiling. Traffic inspection solutions examine network traffic content to check whether or not sensitive information leaks outward from an organization. Wei et al. [31, 32] have developed payload attribution techniques that trace data leaks. Another approach is to use steganography or watermarking to ensure that distributors of the marked files cannot deny their leakage [14],

While traffic inspection solutions perform *post mortem* detection of insider threats, behavior profiling solutions are useful for insider threat prediction. Le and Zincir-Heywood [15] have used a hidden Markov model to capture normal user behavior sequences; insider threat alarms are raised when normal sequence violations are observed. Graph-based anomaly detection methods have been used to discover insider threats [7]. Axelrad et al. [2] have developed a directed acyclic graph representation of a Bayesian network for insider threat detection. However, graph construction is costly and human experts are required to manually provide empirical estimates of probabilities.

Machine learning models [9, 15] such as self organizing maps [13] and C4.5 decision trees [19] have been applied to insider threat detection. While several well-designed supervised models have been used to detect anomalies [5], small numbers of insider threat records present in training datasets prevent supervised models from learning insider threat data patterns. Additionally, in real-world situations, labeled insider threat data is generally not available. As a result, unsupervised learning models should be applied to detect insider threats.

Popular unsupervised learning approaches include the k-means [17] and isolation forest [16] methods. One-class classification has also been used for anomaly detection (e.g., one-class kernel Fisher discriminant analysis [21] and one-class support vector machines [23]). However, these methods implicitly assume that all the training data is normal, which is not appropriate in practice. Thus, few, if any, insider threat detection techniques actually employ unsupervised deep learning.

Autoencoders [10] and variants such as denoising autoencoders [30] and variational autoencoders [12] have been used to detect anomalies. Although many anomaly detection applications claim that their models employ unsupervised learning, they still correspond to one-class classification models because they rely on *a priori* labeling to select only normal data for training. The state-of-the-art deep autoencoding Gaussian mixture model [33] also relies on labeled normal data for training, but its performance is sensitive to contamination by anomalies. In other words, the model is not well suited to insider threat detection without labeled data.

In the case of data leakage and intranet attacks, malicious insider activities tend to manifest themselves as anomalous behavior or abnormal network traffic content for the specific insider. For example, it is normal for a salesman to download price records from a remote sales department server and abnormal for a human resources specialist to do so, but the operation itself is normal in the enterprise system. In fact, insider threat behavior is very complex and it is infeasible to use traditional rule-based approaches and estimation theory for detection [3]. Moreover, collecting a large amount of labeled training data manually is difficult and time-consuming. The state-of-the-art unsupervised insider threat detection approach proposed by Tuor et al. [29] augments a basic deep neural network with long short-term memory to recognize insider threat data with high anomaly scores, but the recall rate is not high enough. In contrast, by relying on multiple autoencoders and true unsupervised learning, the proposed framework estimates the distribution of normal encoded data using a Gaussian mixture model and can identify insider threat data. Indeed, experiments demonstrate that the proposed multi-autoencoder-filtered unsupervised learning model has superior detection performance compared with state-of-the-art baseline models.

3. Multi-Autoencoder Detection Framework

A basic autoencoder is a deep neural network with a symmetric structure (Figure 1). The network comprises two fully-connected-layer parts, encoder and decoder, that do not require supervisory labels. The objective of the network is to reconstruct the input in the output layer. In this feed-forward network, the encoder layers encode the input into the middle code layer, following which the decoder layers decode the code layer into the output. The basic loss function is defined as the reconstruction error between the input and output.

Traditionally, an autoencoder implements a non-linear reduction of high dimensional data [18]. Most unsupervised anomaly detection ap-

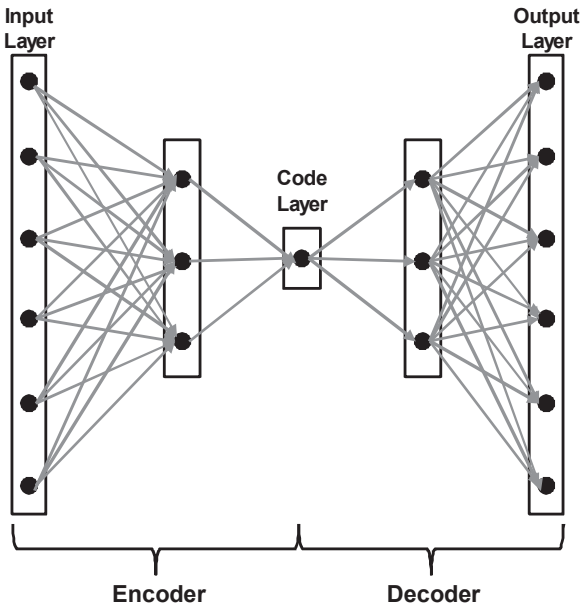

Figure 1. Basic autoencoder structure.

plications that use autoencoders require labeled normal data for training; their models are essentially semi-supervised because they assume that no anomalous data exists in their training sets (which is not realistic). This section presents a true unsupervised learning model that does not need labeled data at any time during its processing.

3.1 Problem Statement

Given a dataset of system logs about user operations without annotated labels, the objective is to detect potential insider threat activity within an organization. The assumption is that insider threat events are rare in system logs. The proposed framework is based on the idea that a deep neural network can learn the patterns of the majority normal data, but it would not reconstruct anomalous data patterns due to the paucity of insider threat data.

Feature aggregation should accommodate five insider threat scenarios [3]: (i) an employee logs in after working hours and uses removable devices to steal sensitive information; (ii) an employee suddenly visits job-hunting websites and emails large attachments to competitors; (iii) an employee masquerades as the employer to send email to employees and disrupt normal company operations; (iv) an employee logs into an-

Table 1. Extracted features.

user, day, role, projects, department, team, supervisor, function, psychometricScoreO, psychometricScoreC, psychometricScoreE, psychometricScoreA, psychometricScoreN, officehour_logon_usualPC, afterhour_logon_unusualPC, officehour_logon_unusualPC, afterhour_logon_usualPC, officehour_deviceConnect, officehour_deviceDisConnect, afterhour_deviceConnect, afterhour_deviceDisconnect, officehour_ FileOpen, officehour_FileCopy, officehour_FileWrite, officehour_FileDelete, afterhour_FileOpen, afterhour_FileCopy, afterhour_FileWrite, afterhour_FileDelete, officehour_unusualUrl_wwwVisit, officehour_usualUrl_wwwVisit, officehour_ unusualUrl_wwwUpload, officehour_usualUrl_wwwUpload, officehour_unusualUrl_wwwDownload, officehour_usualUrl_wwwDownload, afterhour_unusualUrl_wwwVisit, afterhour_usualUrl_wwwVisit, afterhour_ unusualUrl_wwwUpload, afterhour_usualUrl_wwwUpload, afterhour_unusualUrl_wwwDownload, afterhour_usualUrl_wwwDownload, officehour_Logon, officehour_Logoff, afterhour_Logon, afterhour_Logoff, officehour_unusualEmail_AttachYes, officehour_usualEmail_AttachYes, officehour_unusualEmail_AttachNo, officehour_usualEmail_AttachNo, afterhour_unusualEmail_AttachYes, afterhour_usualEmail_AttachYes, afterhour_unusualEmail_AttachNo, afterhour_usualEmail_AttachNo

other employee's computer to find sensitive documents and emails the documents or stores them on a removable device; and (v) an employee suddenly uploads a large number of files to his/her mailbox.

In order to extract appropriate features to distinguish insider threat records from normal records, the log files containing device, email, file, network and login data are combined to aggregate the discriminating features of each record of each user for each day [29]. Event occurrences are recorded in the feature columns (e.g., how many times a user sent email messages with or without large attachments to an unusual third party after office hours in one day).

Table 1 shows the aggregated data and user metadata features after the deletion of meaningless columns. A value is deemed to be usual if it has appeared in more than 5% of the log records before the given log record.

3.2 Multi-Autoencoder Filtering

After extracting appropriate features from the log files, an aggregated feature matrix is constructed. Each row of the aggregated feature matrix corresponds to the operations done by a user during a day and each

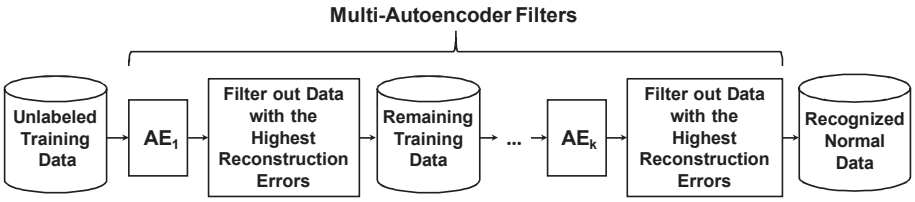

Figure 2. Multi-autoencoder filtering structure.

column represents the number of instances of a specific event or user metadata item. The feature matrix is randomly divided to produce the training and testing datasets.

In the following, the unlabeled training input and the corresponding output are denoted by $X = (x_i^{(j)})$ and $\hat{X} = (\hat{x}_i^{(j)})$, respectively, where $i = 1, \ldots, n$; $j = 1, \ldots, d$; $x_i^{(j)}, \hat{x}_i^{(j)} \in \mathbb{R}$; n is the number of input records; and d is the dimension of the input and output matrices.

Figure 2 shows the multi-encoder filtering structure. The structure comprises k cascaded separately-trained autoencoders with the structure shown in Figure 1. The following steps are involved:

- **Step 1:** Train the first autoencoder (AE_1) using the entire training set.
- **Step 2:** Compute the reconstruction error $L(X, \hat{X})$ between the input X and output $\hat{X}$ according to the following equation:

$$f_i^{(j)} = \frac{\exp(\hat{x}_i^{(j)})}{\sum_{l=1}^{d} \exp\left(\hat{x}_i^{(l)}\right)}, \quad L_i = -\sum_{j=1}^{d} x_i^{(j)} \cdot \log\left(f_i^{(j)}\right) \qquad (1)$$

  where $L(X, \hat{X})$ is an n-dimensional vector whose i^{th} entry is denoted by L_i. Following this, filter out $r\%$ of the training set with the largest reconstruction errors.

- **Step 3:** Train the next autoencoder using the remaining training set. Repeat Step 2 until filtering has been done by all k autoencoders.

Assume that the proportion of the insider events in the entire training set is p_0. Then, the probability of randomly selecting an insider event record is p_0. Let c be a coefficient. Furthermore, let cp_0 be the probability of one autoencoder filtering out one item as an insider threat record. In other words, the ability of one autoencoder to filter insider threat data is c times better than random filtering.

Using the first autoencoder, the corresponding largest $r\%$ of training data is filtered out according to the reconstruction error metric $L(X, \hat{X})$. If the original training dataset contains n total records and s insider threat records, then the proportion of insider records remaining in the training set after the k^{th} autoencoder (AE_k) is p_k.

The remaining proportion of insider threat records in the training set after the first autoencoder (also the same as the original insider threat record proportion before training the second autoencoder) is given by:

$$p_1 = \frac{np_0 - nrcp_0}{n(1-r)} = \frac{1-cr}{1-r}p_0 \tag{2}$$

where $cr \leq 1$.

Mathematical induction yields the following expression:

$$p_k = (\frac{1-cr}{1-r})^k p_0 \tag{3}$$

In order to filter out all the insider threat records in the training set, the following condition must hold:

$$p_k n(1-r)^k < 1 \quad \Rightarrow \quad (1-cr)^k s < 1. \tag{4}$$

The multi-autoencoder filtering process yields data that is almost completely normal with a negligible number of insider threat records. The normal data is then encoded to its code layer representation for the k^{th} autoencoder and the distribution of the recognized encoded normal records is estimated. This enables insider threat records to be identified when their encoded feature representations deviate from the distribution of recognized encoded normal records.

3.3 Insider Threat Prediction

After recognizing the compressed representation of pure normal data, the data is fitted to a Gaussian mixture model to estimate the distribution of normal encoded data.

Let $Z = (z_i^{(j)})$ be the compressed representation of the normal input to the code layer, where $i = 1, \ldots, m$; $j = 1 \ldots, d$; $z_i^{(j)} \in \mathbb{R}$; m is the number of recognized normal records; and d is the dimension of the compression representation in the code layer. Then, the probability density function of the multivariate Gaussian mixture distribution parameterized by $\theta = \left\{(\theta_c = (\mu_c, \Sigma_c^2), \alpha_c)\right\}_{c=1}^{C}$ is given by:

$$P(z \mid \theta) = \textstyle\sum_{c=1}^{C} \alpha_c \Phi(z \mid \theta_c) \tag{5}$$

where C is the number of Gaussian components, (μ_c, Σ_c^2) are the mean and covariance matrix of the c^{th} Gaussian component, α_c is the probability that a sample belongs to the c^{th} Gaussian component, and the probability density of the c^{th} Gaussian component $\Phi(z \mid \theta_c)$ is given by:

$$\Phi(z \mid \theta_c) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(z-\mu)^T \Sigma^{-1} (z-\mu)}{2}\right) \tag{6}$$

Maximum likelihood estimation is employed to obtain the Gaussian mixture model parameters. Given a set of N samples $x_1, \ldots, x_N$ and assuming independent sampling, the likelihood $L(\theta)$ of a fixed parameter θ is given by:

$$L(\theta) = \prod_{j=1}^{m} P(z_j \mid \theta) \tag{7}$$

Hence, the log-likelihood $\log L(\theta)$ is given by:

$$\log L(\theta) = \sum_{j=1}^{m} \log P(z_j \mid \theta) \tag{8}$$

The expectation-maximization algorithm [6] is used to maximize the log-likelihood via the following iterative process:

- **Step 1:** Initialize the parameters $\theta = \left\{\mu_c, \Sigma_c^2, \alpha_c\right\}_{c=1}^{C}$.
- **Step 2:** Repeat Steps 3 and 4 in sequence to update θ until convergence.
- **Step 3:** Compute the probability γ_{jc} that sample j comes from the c^{th} component as follows:

$$\gamma_{jc} = \frac{\alpha_c \Phi(z_j \mid \theta_c)}{\sum_{c=1}^{C} \alpha_c \Phi(z_j \mid \theta_c)}, \quad j = 1, \ldots, m; \quad c = 1, \ldots, C$$

- **Step 4:** Update the parameters:

$$\begin{aligned}
\mu_c &= \frac{\sum_j^m (\gamma_{jc} z_j)}{\sum_j^m \gamma_{jc}}, \quad c = 1, \ldots, C \\
\Sigma_c &= \frac{\sum_j^m \gamma_{jc} (z_j - \mu_c)(z_j - \mu_c)^T}{\sum_j^m \gamma_{jc}}, \quad c = 1, \ldots, C \\
\alpha_c &= \frac{\sum_j^m \gamma_{jc}}{C}, \quad c = 1, \ldots, C
\end{aligned}$$

Given a sample input x, it is compressed to z via multi-autoencoder filtering, after which the negative log probability density, $-\log P(z \mid \theta)$, is computed as its behavior score. All the samples with behavior scores larger than a threshold ε are predicted to be insider threat records. The threshold ε is set based on cross-validation.

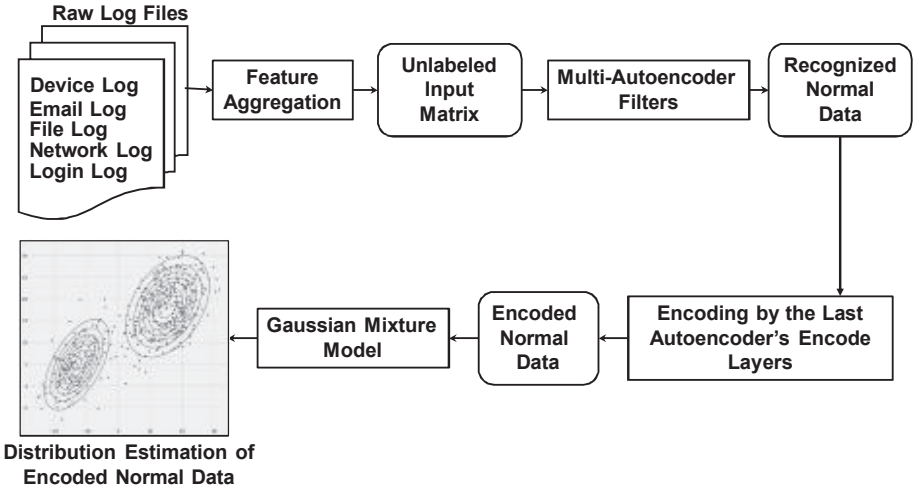

Figure 3. Framework workflow.

3.4 Framework Workflow

Figure 3 shows the overall framework workflow. A total of 53 features were aggregated from the log files of user login/logout activities, operations on devices, files, email and network connections, along with user metadata. The feature matrix was then used to train the multi-autoencoder filtering model in an unsupervised manner without labels. Each row in the matrix corresponded to user behavior on a given day. Multi-autoencoder filtering was used to recognize a portion of the normal data. Meanwhile, the code layer in the last autoencoder was treated as the appropriate low-dimensional representation of normal behavior profiling. Next, the compressed normal data was used to fit a Gaussian mixture model to estimate the distribution of normal encoded data. Records with behavior scores larger than the threshold were predicted to be insider threat records.

4. Framework Evaluation

This section presents the evaluation results obtained when applying the framework to the Insider Threat Dataset (r6.2) [26].

4.1 Multi-Autoencoder Filtering Performance

Figure 4 shows the theoretical minimum k values for different proportions $r = 10\%$, 20% and 30% for dropping insider threat items from the training set based on coefficient c (Equation (4)). The number of insider

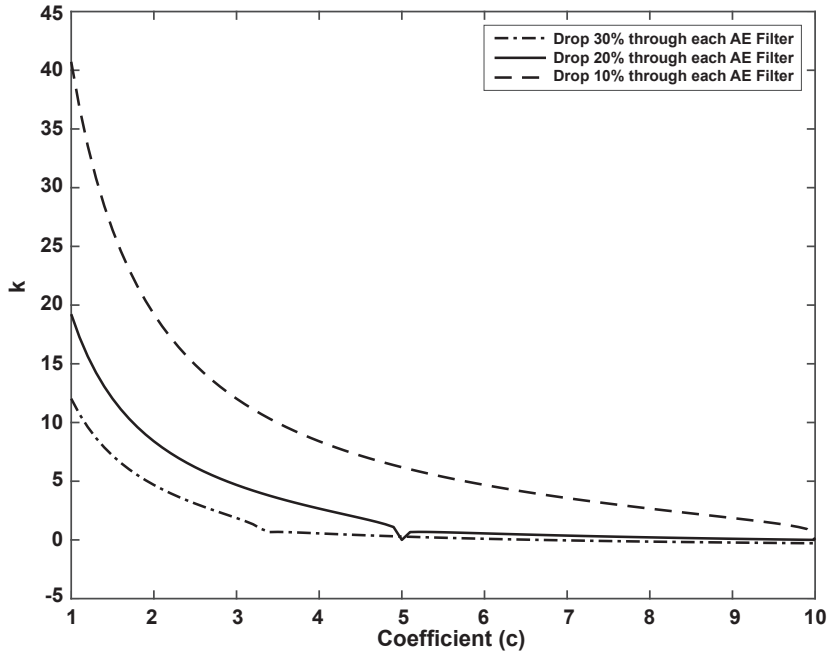

Figure 4. Minimum k values for eliminating all insider threat items.

records in the synthetic dataset after feature aggregation was $s = 73$ out of a total 1,391,247 records.

A key evaluation metric is the purification performance of autoencoder filtering versus that of random filtering. A coefficient c greater than one implies that autoencoder filtering of insider threat records is better than random filtering.

Figure 5 shows the percentages of insider threat records remaining in the training set after one round of autoencoder filtering and random filtering over 100 trials.

Figure 6 shows the percentages of insider threat records remaining in the training set after using five ($k = 5$) multi-autoencoder filters (MAFs) and after five random filtering rounds over 100 trials.

Table 2 shows the corresponding average remaining insider record percentages after multi-autoencoder and random filtering.

Figure 7 shows the filtering performance of five continuous autoencoder filters on the Insider Threat Dataset (r6.2) during a single trial. The graph reveals that the proportion of insider threat records remaining in the training set continuously decreases from autoencoder to autoencoder.

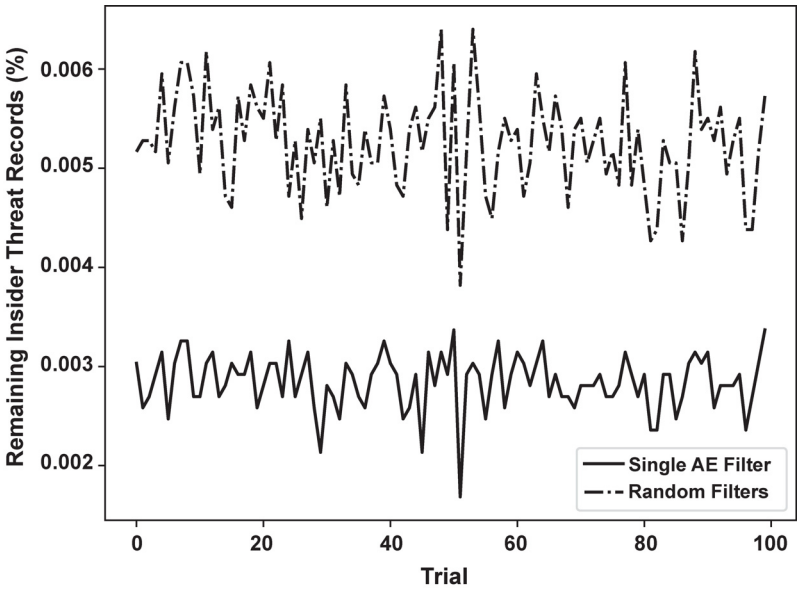

Figure 5. Insider threat records after one round of filtering.

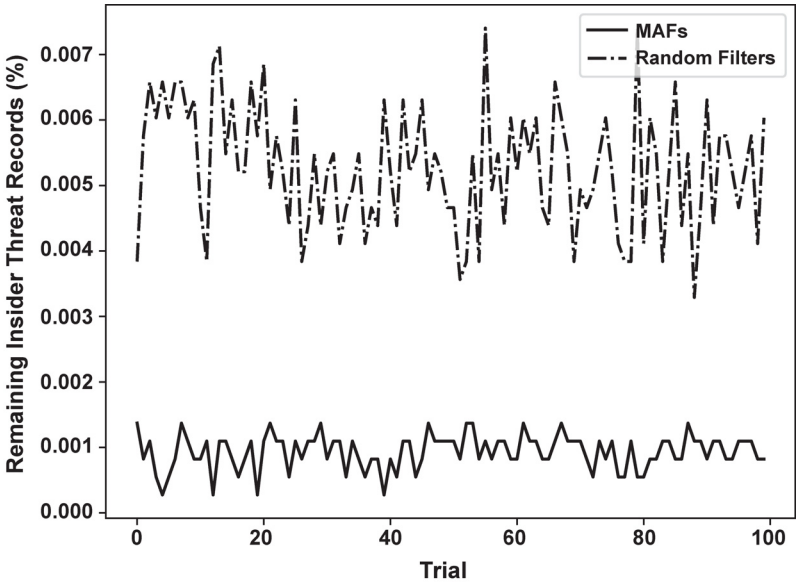

Figure 6. Insider threat records after five rounds of filtering.

Table 2. Average percentages of remaining insider threat records after filtering.

Filtering Technique	Remaining Insider Threat Records	
	One Round	Five Rounds
Autoencoder Filtering	0.0028301982%	0.0009267764%
Random Filtering	0.0052504669%	0.0052508192%

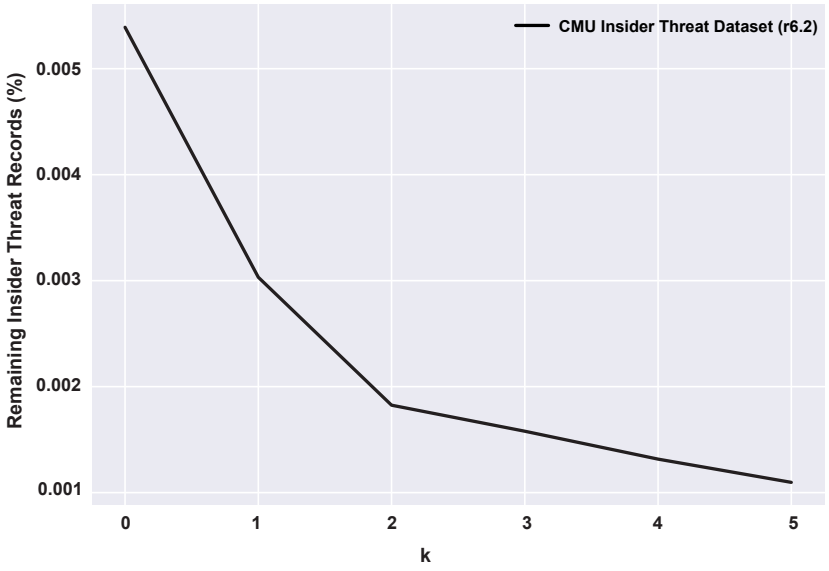

Figure 7. Multi-autoencoder filtering performance in a single trial.

4.2 Comparison Against Baseline Methods

This section compares the performance of the proposed framework against state-of-the-art baseline models, specifically, the k-means, one-class support vector machine, unsupervised deep neural network [29], one-class autoencoder [8] and deep autoencoding Gaussian mixture [33] models using the Insider Threat Dataset (r6.2).

Applying feature extraction to the dataset yielded 53 dimensions of 1,391,247 instances that included only 73 insider threat records. The dataset was randomly split into a training set, cross-validation set and testing set with percentages of 80%, 10% and 10%, respectively. Note that the one-class classification models (i.e., one-class support vector machine, one-class autoencoder and deep autoencoding Gaussian mixture models) were semi-supervised over the entire anomaly detection process. The implicit assumption was that they had a high-quality training set

Table 3. Comparison of the proposed framework against five baseline models.

Method	Recall	AUC
k-means model	0	NA
One-class support vector machine model	0.733	0.405
Unsupervised deep neural network model	0.556	0.625
One-class autoencoder model	0.364	0.589
Deep autoencoding Gaussian mixture model	0.476	0.692
Proposed framework	**0.923**	**0.925**

containing only normal data, although, in a real unsupervised learning scenario, it would not be known if any (and how much) anomalous data existed in the training dataset.

The proposed framework and the other deep learning models were implemented in Keras [11] running on the TensorFlow [1] backend. The k-means and one-class support vector machine models were implemented in scikit-learn [24]. All the experiments were executed on an Intel Core i5-3570 2.4 GHz CPU with 32 GB memory.

The framework was configured with five autoencoders and dropping rate $r = 20\%$. Each autoencoder network executed with Input(53) – Dense(53, 50, none) – Dense(50, 25, tanh) – Dropout(0.2) – Dense(25, 8, relu) – Dense(8, 25, relu) – Dropout(0.2) – Dense(25, 50, tanh) – Dense(50, 53, relu), where Input(x) is an input layer with x-dimensional input, Dense(i, o, g) is a fully connected layer with i input neurons and o output neurons with activation function g, and Dropout(d) denotes a drop out of d% of neurons to avoid overfitting. All the autoencoders were compiled by a stochastic gradient descent optimizer with a learning rate of 1×10^{-4}, training epoch number of 500 and batch size of 1,024.

In insider threat detection scenarios, due to the property that malicious insiders are rare and just one overlooked incident could cause considerable damage, the recall metric is more significant than other metrics. In other words, it is critical to detect the insider threat even if the number of false alarms are increased. At the same time, a good solution would reduce the number of false alarms to the extent possible while maintaining a high recall value. Because recall (also called the true positive rate) and the false positive rate constitute a tradeoff, the area under the ROC curve (AUC) is also used as a metric. Note that the horizontal and vertical axes of the ROC curve correspond to the false positive rate and true positive rate (recall), respectively.

Table 3 compares the performance of the proposed framework against the five baseline models. The proposed framework clearly outperforms all the baseline models. Indeed, the proposed framework has recall and

AUC scores that are more than 19% and 23% higher, respectively, than the best scores of the other five models.

5. Conclusions

The proposed unsupervised deep learning framework for insider threat detection is an advancement over other unsupervised deep learning models that require a training dataset containing labeled normal data and do not work well when the training dataset includes anomalous (i.e., insider threat) data. The framework leverages automated multi-autoencoder filtering to eliminate anomalies and then estimates the distributions of encoded and recognized normal data using a Gaussian mixture model. Data with negative log probability density values larger than a threshold are identified as insider threat data. Experiments demonstrate that the multi-autoencoder-filtered unsupervised learning framework has much better recall and AUC scores compared with five state-of-the-art insider threat detection models.

The framework is founded on the notion that an autoencoder can reconstruct the majority normal data, but cannot reconstruct rare insider threat data satisfactorily. Due to the difficulty of detecting insider threat data using a deep neural network without supervisory labels, the only option is to filter out potentially anomalous data with larger reconstruction errors. However, this approach filters out portions of normal data, which reduces the amount of normal data for estimating the multivariate Gaussian mixture model distribution, contributing to an elevated false positive rate [4]. Future research will modify the framework to decrease the false positive rate.

The current version of the framework is designed for static data. Future research will extend the framework to detect anomalies in sequential and spatial data. Since the encoded non-linear representation of the input is automatically generated by multi-autoencoder filtering, the research will treat concatenated log files as inputs and use natural language processing methods to solve the anomaly detection problem.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, TensorFlow: A system for large-scale machine learning, *Proceedings of the Twelfth USENIX Symposium on Operating Systems Design and Implementation*, pp. 265–283, 2016.

[2] E. Axelrad, P. Sticha, O. Brdiczka and J. Shen, A Bayesian network model for predicting insider threats, *Proceedings of the IEEE Security and Privacy Workshops*, pp. 82–89, 2013.

[3] S. Axelsson, A Preliminary Attempt to Apply Detection and Estimation Theory to Intrusion Detection, Technical Report, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 2000.

[4] S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection, *ACM Transactions on Information and System Security*, vol. 3(3), pp. 186–205, 2000.

[5] V. Chandola, A. Banerjee and V. Kumar, Anomaly detection: A survey, *ACM Computing Surveys*, vol. 41(3), article no. 15, 2009.

[6] A. Dempster, N. Laird and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39(1), pp. 1–38, 1977.

[7] W. Eberle and L. Holder, Applying graph-based anomaly detection approaches to the discovery of insider threats, *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*, pp. 206–208, 2009.

[8] D. Ellison, Fraud detection using autoencoders in Keras with a TensorFlow backend, *Oracle AI and Data Science Blog*, August 9, 2018.

[9] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal and R. Rolleston, Supervised and unsupervised methods to detect insider threats from enterprise social and online activity data, *Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications*, vol. 6(4), pp. 47–63, 2015.

[10] G. Hinton and R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science*, vol. 313(5786), pp. 504–507, 2006.

[11] Keras, Keras API Reference (`keras.io/api`), 2020.

[12] D. Kingma and M. Welling, Auto-Encoding Variational Bayes, *arXiv:* 1312.6114v10, 2014.

[13] T. Kohonen, Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, vol. 43(1), pp. 59–69, 1982.

[14] A. Kumar and K. Pooja, Steganography – A data hiding technique, *International Journal of Computer Applications*, vol. 9(7), pp. 19–23, 2010.

[15] D. Le and A. Zincir-Heywood, Evaluating insider threat detection workflow using supervised and unsupervised learning, *Proceedings of the IEEE Security and Privacy Workshops*, pp. 270–275, 2018.

[16] F. Liu, K. Ting and Z. Zhou, Isolation forest, *Proceedings of the Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[17] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[18] K. Pearson, On lines and planes of closest fit to systems of points in space, *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, vol. 2(11), pp. 559–572, 1901.

[19] J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, Burlington, Massachusetts, 2014.

[20] T. Rashid, I. Agrafiotis and J. Nurse, A new take on detecting insider threats: Exploring the use of hidden Markov models, *Proceedings of the Eighth ACM International Workshop on Managing Insider Security Threats*, pp. 47–56, 2016.

[21] V. Roth, Kernel Fisher discriminants for outlier detection, *Neural Computation*, vol. 18(4), pp. 942–960, 2006.

[22] M. Salem, S. Hershkop and S. Stolfo, A survey of insider attack detection research, in *Insider Attack and Cyber Security*, S. Stolfo, S. Bellovin, A. Keromytis, S. Hershkop, S. Smith and S. Sinclair (Eds.), Springer, Boston, Massachusetts, pp. 69–90, 2008.

[23] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, Support vector method for novelty detection, *Proceedings of the Twelfth International Conference on Neural Information Processing Systems*, pp. 582–588, 1999.

[24] scikit-learn, Machine learning in Python (`scikit-learn.org`), 2019.

[25] G. Silowash, T. Lewellen, J. Burns and D. Costa, Detecting and Preventing Data Exfiltration Through Encrypted Web Sessions via Traffic Inspection, Technical Note, CMU/SEI-2013-TN-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2013.

[26] Software Engineering Institute, Insider Threat Test Dataset, Carnegie Mellon University, Pittsburgh, Pennsylvania (`resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099`), 2016.

[27] Splunk Technology, Splunk, San Francisco, California (`www.splunk.com`), 2020.

[28] Thales Digital Identity and Security, Breached records more than doubled in H1 2018, reveals Breach Level Index, *Thales Digital Identity and Security Blog*, October 23, 2018.

[29] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols and S. Robinson, Deep learning for unsupervised insider threat detection in structured cybersecurity data streams, presented at the *Thirty-First AAAI Conference on Artificial Intelligence Workshop on AI and OR for Socal Good*, 2017.

[30] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[31] Y. Wei, F. Xu, X. Chen, Y. Pu, J. Shi and S. Qing, Winnowing double structure for wildcard query in payload attribution, *Proceedings of the Seventeenth International Conference on Information Security*, pp. 454–464, 2014.

[32] Y. Wei, F. Xu, X. Chen, J. Shi and S. Qing, Winnowing multihashing structure with wildcard query, *Proceedings of the Asia-Pacific Conference on Web Technologies and Applications*, pp. 265–281, 2014.

[33] B. Zong, Q. Song, M. Min, W. Cheng, C. Lumezanu, D. Cho and H. Chen, Deep autoencoding Gaussian mixture model for unsupervised anomaly detection, poster presented at the *Sixth International Conference on Learning Representations*, 2018.