Hindawi
Security and Communication Networks
Volume 2018, Article ID 5906368, 18 pages
https://doi.org/10.1155/2018/5906368

WILEY | Hindawi

Research Article

Distance Measurement Methods for Improved Insider Threat Detection

Owen Lo, William J. Buchanan, Paul Griffiths, and Richard Macfarlane

Edinburgh Napier University, Edinburgh, UK

Correspondence should be addressed to William J. Buchanan; w.buchanan@napier.ac.uk

Received 24 August 2017; Revised 6 December 2017; Accepted 13 December 2017; Published 17 January 2018

Academic Editor: Gerardo Pelosi

Insider threats are a considerable problem within cyber security and it is often difficult to detect these threats using signature detection. Increasing machine learning can provide a solution, but these methods often fail to take into account changes of behaviour of users. This work builds on a published method of detecting insider threats and applies Hidden Markov method on a CERT data set (CERT r4.2) and analyses a number of distance vector methods (Damerau–Levenshtein Distance, Cosine Distance, and Jaccard Distance) in order to detect changes of behaviour, which are shown to have success in determining different insider threats.

1. Introduction

The act of an insider threat involves malicious activity which occurs from within an organization or company and is conducted by a member of such an entity. We propose that a greater interest in this area of research has been noted in recent years due to high profile public cases of individuals who have committed acts of data leakage along with organizations which have been set up specifically to publish such data.

In this work, we demonstrate capabilities of detecting insider threats against a synthetic dataset which is referred to as the CERT Insider Threat Dataset. We apply distance measurement techniques which have been used in the past for text [1, 2] and speech analysis [3, 4] alongside more computer information oriented tasks such as similarity hashing and data mining [5, 6]. We begin by first reviewing related works in this field of research.

For Machine Learning (ML) there are typically two main phases [7]: training and testing, with a common set of steps of defining the features and classes within the training data set. Next a subset of attributes is located for classification, and a learning model is applied on the training data. With the learning model, the rest of the data is then fitted back and the success rate determined. The basic process that we have in applying machine learning to cyber security is as follows [8]:

(i) *Information sources*: this involves defining the sources of information that would be required to capture the right information.

(ii) *Data capturing tools*: this involves creating the software agents required to process the required data.

(iii) *Data preprocessing*: this involves processing the data into a format which is ready for the analysis part.

(iv) *Feature extraction*: this involves defining the key features that would be required to the analysis engine.

(v) *Analysis engine*: this involves the creation of an analysis engine which takes the features and creates scoring to evaluate risks.

(vi) *Decision engine*: this takes the scoring systems from the analysis stage and makes a reasoned decision on the level of risk involved.

SIEM (Security Information and Event Management) tools, such as Splunk, analyse security logs using well-defined schemas, but new log file analysis can be difficult without the required schema. Nimbalkar et al. [9] thus define a framework which uses a semantic description of a schema and content in RDF (Resource Description Framework). This activity is typically defined in the form of user activity, network traffic, process activity, and so on. Their method involves normalising the security logs into columns and rows

and then uses regular expression-based and dictionary-based classifiers. These are then used to map into domain-specific ontologies (such as the Unified Cybersecurity Ontology [10]) and more general ones (such as DBpedia [11]).

The two main approaches used within the detection of threats is within signature detection, where we match against well-known patterns of malicious behaviour, or anomaly detection, where we define a normal behaviour pattern, and then detect deviations away from this. For machine-focused threats, such as for viruses and worms, the signature methods are best, but for the detection of human focused zero-day threats and for others, such as for fraud and data theft, the anomaly detection methods normally work best [7].

Within machine learning there are often two main phases: training and testing. This involves definition of attributes (features) and classes from training data and then reducing these to a subnet which can be used within the classification process. A model is then created with the training set, where the model is then used on the complete data set in order to understand the success rate. This output may be in the form of a metric or a decision. When it is a decision we typically define a confusion matrix approach where we measure the predicted values against the actual values.

Within a decision engine, we often use the concept of correct guesses (true) and incorrect ones (false). So a true positive is where we determine that an event was correctly detected, while a false positive is where a true event was not detected (and thus missed by the system). Within IDS (Intrusion Detection Systems) there is often a balance to be struck when tuning the systems so that users do not get swamped by too many false alerts (false-positives) or from too many fake alerts.

2. Related Works

Recent work in the application of data analytic techniques against the CERT dataset include [12–15]. The work of [13] focuses on the CERT r6.2 dataset and uses Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN) to calculate an anomaly score of each individual user. Their results demonstrate that insider threat activity, using their implementation, would produce an average score in the 95.53 percentile range.

The work of [12] applies another well-known machine learning technique, Hidden Markov Models, to the detection of insider threats. This work focuses on the CERT r4.2 dataset and it demonstrates a case study against malicious user "MCF0600" that produces a low value for negative log likelihood during the week malicious activity that occurs.

In [14], the authors of this research implement a framework named RADISH and demonstrate capabilities in using both k-Nearest Neighbour (k-NN) and k-Dimensional Tree (k-d tree) in detection of insiders against the CERT r2 dataset. Their results show that k-NN is generally more accurate and faster in comparison with k-d tree. Additionally, the work of [15] uses both the CERT dataset and bespoke synthetic dataset generated to evaluate their approach which is based on grouping of users based on role and applying Principal Component Analysis (PCA) to detect outliers in user activity.

Other related works, which do not use the CERT dataset but are applicable to this research area, include [16] which also uses the k-NN algorithm against access logs. In this work, deviation from normality is the metric used to detect insider threats which is derived by comparing resource access activity of users against each other (the higher the deviation is, the more likely the user is performing malicious activity). Lastly, the work of [17] makes use of physiological signals (EEG and ECG) and demonstrates capabilities of differentiating between malicious and normal users during the act of insider threats.

The work presented in this paper differs from the related works described as we focus on applying distance measurements against the CERT dataset in place of more 'heavy' weight algorithms such as Neural Networks, Deep Learning [13], and HMM [12]. Traditionally, distance measurements have been applied to text analysis [1, 2] and speech based matching [3, 4]. Our work most closely matches classification techniques such as k-NN and PCA as applied by [14] and [15], respectively. However, in the case of our distance measurement implementations, we only consider one set of prior data rather than attempt to classify 'normal' and 'anomalous' behaviour based on the whole range of available data. Thus, we hypothesise in this work that insider threat activity is nondeterministic and learning of extended prior activity does not always enable an increased accuracy in the prediction of a malicious user.

Our methodology for extrapolation of data from the CERT dataset is derived from [12] where the same approach was used for evaluation against HMM. Thus, since our dataset will be presented in the same structure as the work of [12], we chose to benchmark our distance measurement results against the HMM technique to enable a more meaningful evaluation to take place where we assess the effectiveness of our proposed techniques against previous work. Full evaluation against every single other machine learning technique is currently out of the scope of this paper but may be considered in future work. The next section first gives overview of the CERT dataset.

3. CERT Dataset

This section provides an overview of the CERT r4.2 dataset which is used for our evaluation of distance measurements in the detection of malicious users.

3.1. Overview. The CERT dataset [18] consists of synthetic data mixed in with benign data. We focus on release version r4.2 of the dataset within the scope of this paper. The r4.2 dataset contains both benign and malicious user activity. There are 7 primary groups of files which are generated from 1000 simulated users. A description on the contents of each file is provided in Table 1; further details can be obtained from the CERT website at [19]. In terms of insider threats, version r4.2 of the dataset consists of three primary scenarios described as follows:

(1) User who did not previously use removable drives or work after hours begins logging in after hours, using a

Table 1: CERT r4.2 file description.

Filename	Description
device.csv	Connection and disconnection of removable devices (e.g., USB hard drive) is described in this file.
email.csv	Contains logs of user emails.
file.csv	File access activity is provided in this file.
logon.csv	Relates to user activity based on logging on and logging off on computing devices.
psychometric.csv	Provides personality and job satisfaction variables for each of the 1000 simulated users.
LDAP	This folder contains a set of LDAP files which describe the ontology of each simulated user (their role, email, department, supervisor, etc.).

removable drive, and uploading data to wikileaks.org and leaves the organization shortly thereafter.

(2) User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.

(3) System administrator becomes disgruntled and downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

In this paper, our focus is on capabilities of detecting anomalous insiders using distance measurement techniques through analysis of only user activity. In other words, we do not take into consideration the personality of each simulated user, nor the contents of each activity, nor the ontology of the users. Thus, our focus is on extrapolation of data from the files device.csv, email.csv, file.csv, and logon.csv only. We have chosen to focus on the CERT 4.2 dataset as our data extrapolation methodology is derived from the existing work of [12]. Additionally, the CERT r4.2 dataset contains a high number of insider threats (in comparison with previous datasets) which is well suited to this work since our goal is to assess the viability of distance measurements in comparison with the HMM approach. The data extrapolation is described in the following section.

3.2. Data Extrapolation of CERT Dataset. The methodology applied to data extrapolation is derived from the work of [12]. For each of the four files of interest (device.csv, email.csv, file.csv, and logon.csv), we parse the file and retain the username, date and time of the activity, and the string description of the activity. Next, for the string description of the activity we replace this with a numeric value ranging from 1 to 7 for ease of analysis in the later stages of this work. Table 2 provides details on the activity and its corresponding numerical label. Finally, each entry is labelled with the week

Table 2: Numeric labels for activities.

Activity	Numerical label
User logs on to a device.	1
User logs off a device.	2
User connects a removable device.	3
User disconnects a removable device.	4
User browses the web.	5
User sends or receives email.	6
User performs file manipulation tasks.	7

Filter

	X1	date	user	activity	week
	All		All	All	All
1	1	2010-01-02 06:49:00	NGF0157	1	1
2	2	2010-01-02 06:50:00	LRR0148	1	1
3	3	2010-01-02 06:53:04	LRR0148	1	1
4	1260240	2010-01-02 06:55:16	LRR0148	5	1
5	4	2010-01-02 07:00:00	IRM0931	1	1
6	5	2010-01-02 07:00:00	MOH0273	1	1
7	1260241	2010-01-02 07:00:13	NGF0157	5	1
8	1260242	2010-01-02 07:03:46	NGF0157	5	1
9	1260243	2010-01-02 07:05:26	IRM0931	5	1
10	1260244	2010-01-02 07:05:52	IRM0931	5	1

Showing 1 to 11 of 32,770,222 entries

Figure 1: Parsed result of CERT CSV files.

in which the activity occurred (relative to the first log entry). The parsed result of the first 10 entries is shown in Figure 1.

The source code of this data extrapolation is provided in Algorithm 1. The extrapolation process, and other examples in this paper, were performed using the R programming language [20]. The R packages "lubridate" [21] and "readr" [22] were used for date/time related tasks and reading of coma delimited files.

4. Analytic Techniques

In this section, we describe the techniques we chose to implement in this work including HMM, Damerau–Levenshtein (DL) Distance, Jaccard Distance, and Cosine Distance. The HMM analysis technique is a replication of results originally from the work of [12] while the distance measurement techniques, in its application against the CERT data, forms our contribution in this paper.

4.1. Hidden Markov Models. A Markov model is a stochastic system which determines the probability of events based on the current state of the system. A Hidden Markov Model (HMM) works on the same concept with the exception being that the state of the system is only partially observable (as opposed to complete transparency in a Markov model). To calculate probability of events, HMM will monitor any observable outputs of a system to determine probabilities. As defined by [23]:

> *"An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed*

```
library(lubridate)
library(readr)
#Read Each File ##############################
logon <- read_csv("~/r4.2/logon.csv", col_types = cols(id = col_skip(), pc
= col_skip()))
device <- read_csv("~/r4.2/device.csv", col_types = cols(id = col_skip(),
pc = col_skip()))
http <- read_csv("~/r4.2/http.csv", col_types = cols(content = col_skip(),
id = col_skip(), pc = col_skip(), url = col_skip()))
email <- read_csv("~/r4.2/email.csv", col_types = cols(attachments =
col_skip(), bcc = col_skip(), cc = col_skip(), content = col_skip(), from
=col_skip(), id = col_skip(), pc = col_skip(), size = col_skip(), to =
col_skip()))
file <- read_csv("~/r4.2/file.csv", col_types = cols(content = col_skip(),
filename = col_skip(), id = col_skip(), pc = col_skip()))
http["activity"] = "Http"
email["activity"] = "Email"
file["activity"] = "File"
###############################################
#Assign Tags to Each Activity ################
# Logon = 1, Logoff = 2, Connect = 3, Disconnect = 4, Http = 5, Email = 6, File = 7
logon$activity = replace(logon$activity, logon$activity=="Logon", 1)
logon$activity = replace(logon$activity, logon$activity=="Logoff", 2)
device$activity = replace(device$activity, device$activity=="Connect", 3)
device$activity = replace(device$activity, device$activity=="Disconnect", 4)
http$activity = replace(http$activity, http$activity=="Http", 5)
email$activity = replace(email$activity, email$activity=="Email", 6)
file$activity = replace(file$activity, file$activity=="File", 7)
###############################################
#Data Frame Conversion and Join ##############
logon <- as.data.frame(logon)
device <- as.data.frame(device)
http <- as.data.frame(http)
email <- as.data.frame(email)
file <- as.data.frame(file)
join <- mapply(c, logon, device, http, email, file, SIMPLIFY=FALSE)
join <- as.data.frame(join)
###############################################
#Parse and Sort data by Date/Time ############
join$date <- as.POSIXct(join$date, format = "%m/%d/%Y %H:%M:%S")
join <- join[order(join$date),]
join$week <- (as.numeric(join$date-join$date[1]) %/% 604800) + 1 #Label the
weeks starting at week 1
###############################################
```

Algorithm 1: CERT data extrapolation code.

> *through another set of stochastic processes that produce the sequence of observed symbols."*

In machine learning, HMM are used in numerous areas of learning including the detection of insider threats. A recent paper by [12] has demonstrated the capabilities of this technique in the detection of insider threat through experimentations against the CERT dataset (dataset r4.2) [19]. The work of [12] involved training their Hidden Markov Model against the first five weeks of each individual user before applying the model against a user's future activity.

HMM uses probability to determine an unknown set of sequences through the observation of a known set of sequences. More specifically, a HMM can be used to solve three primary problems:

(1) Given a set of sequences, and the computed Hidden Markov model, what is the probability of this sequence?

(2) Given a set of sequences, how do we define a model which best fits this sequence?

(3) How do we adjust the model parameters to best fit a sequence?

In the application of HMM against the CERT dataset, the work of [12] has applied the first and third solutions to detect

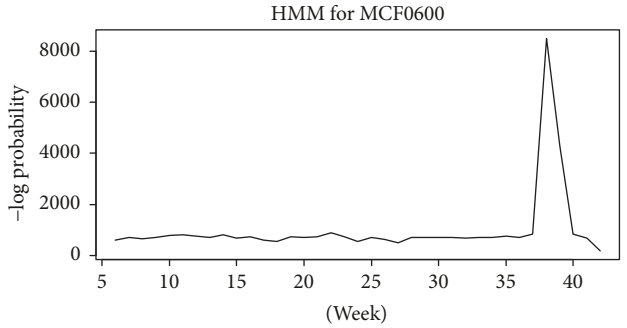
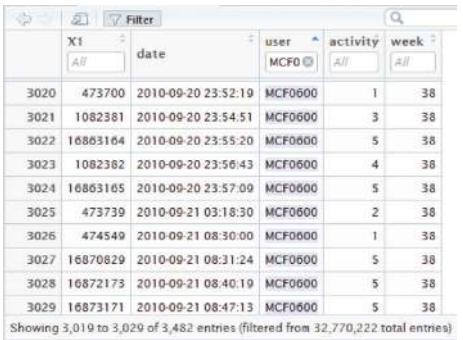


Figure 2: HMM result for CERT user MCF0600 based on methodology by [12].



	X1	date	user	activity	week
	All		MCF0	All	All
3020	473700	2010-09-20 23:52:19	MCF0600	1	38
3021	1082381	2010-09-20 23:54:51	MCF0600	3	38
3022	16863164	2010-09-20 23:55:20	MCF0600	5	38
3023	1082382	2010-09-20 23:56:43	MCF0600	4	38
3024	16863165	2010-09-20 23:57:09	MCF0600	5	38
3025	473739	2010-09-21 03:18:30	MCF0600	2	38
3026	474549	2010-09-21 08:30:00	MCF0600	1	38
3027	16870829	2010-09-21 08:31:24	MCF0600	5	38
3028	16872173	2010-09-21 08:40:19	MCF0600	5	38
3029	16873171	2010-09-21 08:47:13	MCF0600	5	38

Showing 3,019 to 3,029 of 3,482 entries (filtered from 32,770,222 total entries)

Figure 3: Malicious Activity for user "MCF0600".

insider threats. Their work has shown that by learning prior weeks of each individual user's activity (with the assumption that the first five weeks of data for each user in the CERT dataset consist of benign activity) and applying the first solution, one is capable of producing a likelihood of malicious activity based on prior activity. A version of their work (no random restarts to maximise the likelihood of a sequence) was replicated for this paper and proves to be valid for the scenario they depicted against user "MCF0600" where a peak in the negative log probability correlates with the insider threat for this user (Figure 2).

One difference which should be noted is that our implementation here shows that the insider threat occurs at Week 38 rather than Week 39 as described in [12]. The reason for this is attributed to the fact that we may have processed the date time (relative to a week) in a different manner. However, inspection of our parse dataset does demonstrate that Week 38 contains user activity for the dates 20/9/2010 until 23/9/2010 for user "MCF0600" and correlates with when the insider threat occurred as defined by the CERT answer file. The first 10 entries of where insider threat activity begins for "MCF0600" are shown in Figure 3 as evidence that our week label matches the threat.

Although the work of [12] has demonstrated capability in using Hidden Markov Models (HMM) for detection of insiders in the CERT dataset, it was acknowledged and observed that the computation time to generate a Hidden Markov Model (i.e., the training phase) can be quite slow as the number of data variables increases. Furthermore, deeper

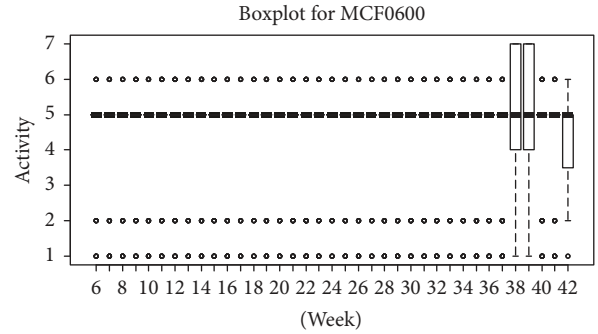


Figure 4: Box plot for user MCF0600.

analysis of the user MCF0600 demonstrates that this simulated users activity is uniform until the very week in which insider threat occurs as shown in the box plot in Figure 4 which may mitigate the need for more complex learning algorithms. For every single week, the box plot shows that the median activity for MCF0600 is activity 5 (browsing the web). The only variation in user activity begins at Week 38 which is the week where simulated malicious activity occurs. Additionally the final week of activity, Week 42, produces some variation in activity. This is due to the far lower number of activities which occur in this final week in comparison with previous weeks. We found that the final week of the majority of users demonstrated this trend.

In the subsections which follow, we demonstrate distance measurement techniques which show potential in the detection of insider threats within the CERT r4.2 dataset which works both faster and more efficiently in comparison with HMM.

The source code for producing the HMM result using the R programming language is provided in Algorithm 2 while the source code for the box plot is in Algorithm 3. The R package "HMM" was used for Hidden Markov Model related computations [24].

4.2. Distance Measurements. In the past, distance measurements have been applied to text [1, 2] and speech based matching [3, 4]. More recently, the use of such techniques in cryptography and data mining has been of value, namely, similarity hashing techniques which allow for approximate matching of data [5, 6]. A wide variety of algorithms have been proposed in this area of research (a description of each is outside the scope of this paper); however, the fundamental aspect of each algorithm remains the same: how similar or dissimilar two or more sets of data are given a distance metric.

Given the fact that the CERT r4.2 dataset consists of user activity which may be described as discrete values (ranging from 1 to 7 based on the extrapolation technique from an earlier section of this paper), we believe there is capability of using distance measurements to assess and detect anomalous behaviour which may be identified as insider threats. Within the scope of the work carried out in the remaining sections of this paper, we focus on three distance measurement techniques including Damerau–Levenshtein Distance, Cosine Distance, and Jaccard Distance.

```
library(HMM)
library(readr)
############## Data Parsing Phase ##############
cert_r4_2_dataset <- read_csv("~/cert_r4.2_dataset.csv") #Load the dataset.
Remember to change the path to file location on own machine.
username = "MCF0600"
allWeeks <- split(cert_r4_2_dataset[cert_r4_2_dataset$user %in%
username,]$activity, cert_r4_2_dataset[cert_r4_2_dataset$user
%in% username,]$week) #Filter dataset to only include data relevent to chosen user.
indx <- sapply(allWeeks, length) #Convert the allWeeks variable into DataFrame.
res <- as.data.frame(do.call(cbind,lapply(allWeeks, 'length<-',max(indx))))
################################################
############## HMM Phase ########################
hmm = initHMM(c(1,2,3,4,5,6,7,8,9,10), c(1,2,3,4,5,6,7)) #Initiate a 10
state HMM with 7 labels (which represent activities of user.)
model = baumWelch(hmm, na.omit(unlist(res[1:5])), maxIterations=20,
pseudoCount =0.1, delta = 0.01) #Train our model with the first 5 weeks of user activity.
vector = c()
for (i in 6:length(res)) #For the remaining weeks of activity...
{
#Calculate probability of a given observed sequence with respect to our model
  logForwardProbabilities = forward(model$hmm, na.omit(unlist(res[i]))) #
... calculate the probability of week i occurring against model...
  like <- ((logForwardProbabilities))
  lenthOfLike <- (length(like)/10)
  answer <- sum(like[,lenthOfLike])
  vector[i - 5] <- answer #... store result of probability in vector...
  model = baumWelch(model$hmm, na.omit(unlist(res[1:i])), maxIterations=20,
pseudoCount =0.1, delta=0.01) #... and update model with week i.
 }
######### Plot Result #########
plot(6:(length(res)), vector[1:(length(vector))] * -1, type="l",
xlab="Week", ylab="-Log Probability", main=paste("HMM for", username, sep=" "))
```

Algorithm 2: HMM implementation code.

```
library(readr)
cert_r4_2_dataset <- read_csv("~/cert_r4.2_dataset.csv") #Load the dataset.
username = "MCF0600"
allWeeks <- split(cert_r4_2_dataset[cert_r4_2_dataset$user %in%
username,]$activity, cert_r4_2_dataset[cert_r4_2_dataset$user
%in% username,]$week) #Filter dataset to only include data relevant to chosen user.
indx <- sapply(allWeeks, length) #Convert the allWeeks variable into DataFrame.
res <- as.data.frame(do.call(cbind,lapply(allWeeks, 'length<-',max(indx))))
boxplot(res[6:length(res)], main=paste("Boxplot for", username,
sep=" "), xlab="Week", ylab="Activity")
```

Algorithm 3: Box plot code for user MCF0600.

(1) Damerau–Levenshtein Distance. The DL (Damerau–Levenshtein) Distance, which has been traditionally applied to string values, defines the minimum number of operations required for one value to be changed to the other value. The operations allowed in the DL algorithm include insertion, deletion, and replacement of values. Furthermore, transposition of adjacent values is included too. As an example, the strings 'ca' and 'abc' would have a Damerau–Levenshtein Distance of 2 since 'ca' -> 'ac' -> 'abc'. In other words, a transposition (swap 'c' with 'a') and insertion operation (insert 'b') is required.

The DL Distance, in most examples given, is generally applied to string values but this technique, and other distance techniques which follow, can also be applied directly to integer values. For example, the integer sequence $\{3, 1\}$ applied against $\{1, 2, 3\}$ would also result in a distance metric of 2 since $\{3, 1\}$ -> $\{1, 3\}$ -> $\{1, 2, 3\}$. Thus, we have capabilities to apply this distance measurement technique to our insider

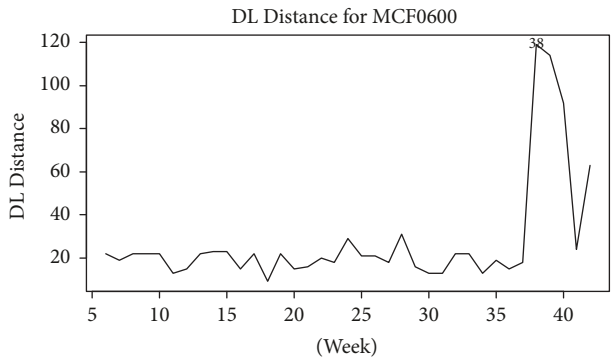


Figure 5: Damerau–Levenshtein Distance for user MCF0600.

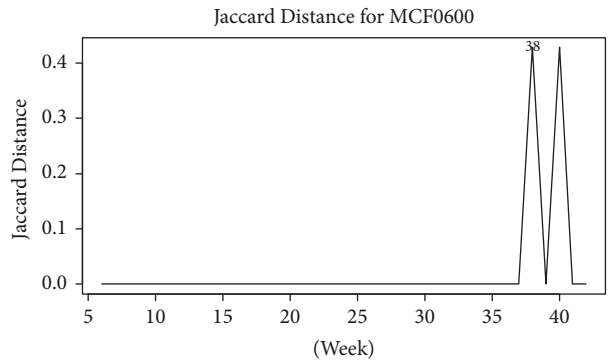


Figure 6: Jaccard Distance for user MCF0600.

threat dataset since our activity values are discrete from range 1 to 7. The equation for DL Distance is presented as follows:

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j), \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1 \quad (a_i \neq b_i) \end{cases} & \text{otherwise}, \end{cases} \tag{1}$$

where a and b are our two sets of data and i and j are the length of each respective sets of data.

Figure 5 presents the DL Distance result for user MCF0600. The labelled data point (Week 38) demonstrates the highest distance from prior activity and correctly matches the week in which the insider threat occurred. The distance metric is calculated as the current week against the prior week (i.e., distance between Week 38 against Week 37).

The R package 'stringdist' [25] was used for calculation of the DL Distance. The source code for this implementation is provided in Algorithm 4. The 'stringdist' package features numerous other algorithms for calculation of distance between values. We investigate some of the distance measurement techniques which are of interest in analysis and detection of insider threats in the CERT r4.2 dataset in the next section.

(2) Jaccard Distance. Jaccard similarity coefficient is a measure of similarity between two sets of data. The similarity coefficient is calculated by dividing the number of shared values (in both sets of data) against the total number of shared and unshared values in both sets of data. As an example, consider the two sets of integers {3, 1} and {1, 2, 3}, the *Jaccard similarity coefficient* would be 0.66 since two integers are the same and there are three shared and unshared values in total (2/3). The *Jaccard Distance* metric measures the dissimilarity between two sets of data and is calculated by simply subtracting 1 from the coefficient value [26]. Thus, our Jaccard Distance for the integer sets {3, 1} and {1, 2, 3} is 0.33. The greater the distance value, the more dissimilar the two sets of data are. The equations for Jaccard coefficient and distance are presented as follows:

$$\text{Jaccard Coefficient} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$
$$\text{Jaccard Distance} = 1 - \text{Jaccard Coefficient}, \tag{2}$$

where A is the first set of variables to be compared against B, the second set of variables.

Figure 6 presents the result of MCF0600 when we calculate the Jaccard Distance for each consecutive week of user activity. Similar to previous results, we see a spike in anomalous activity at Week 38 which correlates with when the insider attack occurred. The result here is produced using the "stringdist" [25] package using the same source code as shown in Algorithm 4 with the method option switched to 'jaccard' instead of 'dl'.

(3) Cosine Distance. Cosine similarity is a measurement of similarity between two sets of data relative to the angle of Cosine between the two datasets. A similarity value of 1 means the two sets of data are the same. In the "stringdist" [25] implementation, Cosine similarity is calculated by first counting the number of occurrences in which each value is observed for both sets of data. Next, the sum of the product of the two sets of counts is then divided by each set of normalised counts multiplied together. An example of calculating Cosine similarity using the data {1, 2, 3, 4, 5} and {1, 2, 3, 4, 4} follows:

(1) The count of occurrences would be {1, 1, 1, 1, 1} and {1, 1, 1, 2, 0}, respectively (the last value is 0 since 5 is not observed in the second set of data).

(2) The sum and product of these two counts would result in the value of 5.

(3) The square root and normalised first set of occurrences is calculated to be 2.24.

(4) The square root and normalised second set of occurrence is calculated to be 2.65.

(5) Multiplying the results of steps (3) and (4) results in 5.92.

(6) Finally, we divide 5 (from step (2)) over 5.92 (step (5)) to get to produce a Cosine similarity value of 0.84.

```
library(readr)
library(stringdist)
cert_r4_2_dataset <- read_csv("~/cert_r4.2_dataset.csv") #Load the dataset.
Remember to change the path to file location on own machine.
username = "MCF0600"
allWeeks <- split(cert_r4_2_dataset[cert_r4_2_dataset$user %in%
username,]$activity, cert_r4_2_dataset[cert_r4_2_dataset$user
%in% username,]$week) #Filter dataset to only include data
relevent to chosen user.
indx <- sapply(allWeeks, length) #Convert the allWeeks variable
into DataFrame.
res <- as.data.frame(do.call(cbind,lapply(allWeeks, 'length<-',max(indx))))
########## Distance Calculation #############
w <- c()
for (i in 6:length(res))
{
  if (i <= length(res))
  {
    di <- seq_dist(na.omit(res[i]), na.omit(res[i-1]), method="dl")
    w[i - 5] <- di
  }
 }
highestDist = 0;
for (result in w)
{
  if ((result) > highestDist)
  {
    highestDist = result
  }
 }
hd_week = match(highestDist, w) + 5
plot(6:(length(res)), w[1:(length(w))], type="l", xlab="Week",
ylab="DL Distance ", main=paste("DL Distance for", username, sep=" "))
text(x=hd_week, y=highestDist, label=hd_week)
```

Algorithm 4: Damerau–Levenshtein distance calculation code.

With a value of 0.84, the Cosine similarity of the two example sets of data can be considered similar to each other (the closer to 0 the more similar they are). In *Cosine Distance*, the value 1 is simply subtracted from Cosine similarity to provide us with an inverse result (i.e., the greater the distance value, the more dissimilar two sets of data may be considered). The equation for Cosine similarity and distance metric is presented next for reference:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3}$$

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity},$$

where A and B are our two sets of respective data.

Figure 7 provides the Cosine Distance result for user MCF0600 when we calculate the current week of the users activity against their previous week. Similar to the Damerau–Levenshtein Distance and Jaccard Distance result, the week where insider attack occurs (Week 38) demonstrates the highest level of Cosine Distance. The result here is produced using the "stringdist" [25] package using the same source code as shown in Algorithm 4. The method is set to 'cosine'.

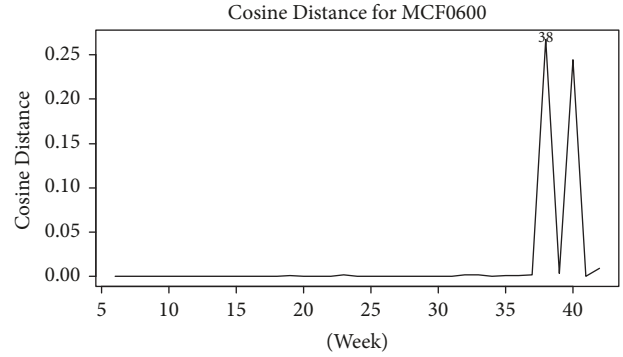


Figure 7: Cosine Distance for user MCF0600.

4.3. Summary. We have demonstrated in this section that HMM and distance measurement techniques both have capability of detecting an insider threat when applied against malicious users within the CERT r4.2 dataset. The results

presented up to this point have focused specifically on user "MCF0600" as a form of baseline to validate whether each of the techniques has capability to detect insider threat based on analysis of user activity. In the section which follows, we present full evaluation of HMM and each of the three distance measurement techniques against all known attackers within the CERT r4.2 dataset. The goal of this evaluation is to provide a comparative benchmark on the ability of each technique in its capability to detect insider threats.

5. Evaluation

5.1. Design. To evaluate the effectiveness of HMM and the three distance measurement techniques described in this work, we apply each algorithm to all known malicious users in the CERT r4.2 dataset. In total, there are 70 malicious users. The insiders are known due to the answer file (provided by CERT) which forms our ground truth for evaluating whether the correct activity is detected as a threat for each malicious user.

Prior to conducting the experiment, we wished to assess whether the techniques we applied would answer the following questions: (1) Would certain techniques better detect certain malicious activity scenarios? and (2) Which technique, overall, would produce the highest detection rate? Answering the first question would be of value in assessing whether certain distance measurement techniques are better at detecting specific types of malicious activity (i.e., Scenario 1, 2, or 3 as described earlier) while answering the second question, naturally, demonstrates how effective the algorithm is in detecting insider threats overall.

5.2. Implementation. For each of the 70 malicious users within the CERT r.4.2 dataset, we first query their related answer file and extract all the dates in which malicious activity is stored. We calculate the distance measurement of user activity by comparing Week_{n+1} against Week_n. Each distance result is stored until all weeks have been analysed for each individual user. For each distance measurement result, we retain the highest value and correlate it with the week and date time in which the user activity occurred. Finally, the date/time value with the highest distance is compared against the answer file provided by CERT to see if there is a match. In this experiment, a match signifies that malicious activity was detected for the correct week of user activity while no match means no malicious activity was detected. The implementation source code for distance measurements is given in Algorithm 5.

For HMM, we apply the methodology as originally described by [12] in which each users activity is first trained on a week-by-week basis and the negative log likelihood of activity for the current week is computed. We retain the week in which the lowest log likelihood value is noted and aim to correlate this with a week found in the malicious users answer file. If a match occurs, we can determine that HMM is capable of detecting malicious activity for the chosen user while no match implies it failed. The implementation source for HMM is given in Algorithm 6. The results of this evaluation are provided in the next section.

Table 3: Detection rate for each distance measurement technique.

	DL	Jaccard	Cosine	HMM
Scenario 1	9	18	12	18
Scenario 2	16	1	20	28
Scenario 3	2	6	1	2
Detection rate	0.39	0.36	0.47	0.69

Table 4: Total detection rate of combined distance measurements.

	DL, Jaccard, and Cosine
All true-positives	56
Total detection rate	0.8

5.3. Results. The detection ratio (number of malicious user detection cases over the total number of malicious users) is presented in Figure 8 while Table 3 gives the number of true-positives per scenario for each technique. Additionally, Table 4 gives the total detection rate of insider threats when combining all three distance measurement techniques. Analysis and discussion of results follow in the next section.

6. Results and Discussion

From the individual detection capabilities of each scenario (Table 3), both Jaccard Distance and HMM demonstrated the greatest capability in the detection of Scenario 1, while HMM was best suited in detection of Scenario 2. Jaccard Distance also scores highest once again in the detection of Scenario 3. For overall detection rate (i.e., the number of insiders detected regardless of scenario), the HMM technique produces the highest detection ratio at 0.69 (48/70) while Jaccard Distance produces the lowest detection ratio of the three techniques tested at 0.35. For each individual scenario, Jaccard Distance and HMM have the greatest capability in the detection of Scenario 1 with 18 true-positives (out of 30 in total) while HMM has the greatest capability in detection of Scenario 2 with 28 true-positives (out of 30 in total). Lastly, Jaccard Distance one again has the greatest capability in detection of Scenario 3 with 6 true-positives out of a total of 10.

From the results presented, one may prematurely conclude that HMM is considered the best of the four techniques demonstrated in this work due to possessing the highest detection rate but deeper analysis of the full individual results (see Table 5) shows that there is still value in the distance measurement techniques. Firstly, we note that each of the three distance measurement techniques demonstrates capability of uniquely detecting a specific malicious user when the other two techniques fail. In the case of DL, 5 unique insiders were detected (examples include 'DCH0843', 'GHL0460', and'KLH0596') while no true-positives were raised by the other techniques.

For Jaccard Distance and HMM, three unique insiders were detected while Cosine Distance resulted in two unique detection instances. Secondly, from a computational perspective, each of the three distance measurement techniques are exceptionally faster than the use of HMM. We were able to

```
library(readr)
library(stringdist)
usernames <- c()
scenarios <- c()
filenames <- c()
dlResults <- c()
jacResults <- c()
cosResults <- c()
files <-
  list.files(
    path = "~/answers",
    pattern = "*.csv",
    full.names = T,
    recursive = TRUE
  )
for (fin files)
{
  username <- regmatches(f, regexpr("-[A-Za-z0-9]+\\.", f))
  username <- sub("-", "", username)
  username <- sub("\\.", "", username)
  usernames <- c(usernames, username)
  scenario <- regmatches(f, regexpr("-[1-3]-", f))
  scenario <- sub("-", "", scenario)
  scenario <- sub("-", "", scenario)
  scenarios <- c(scenarios, scenario)
  filenames <- c(filenames, f)
 }
for (i in 1:length(usernames))
{
  dlResults_temp <- c()
  jacResults_temp <- c()
  cosResults_temp <- c()
  answerFile <-
    read_csv(filenames[i],
             col_names = FALSE,
             col_types = cols_only(X3 = col_guess()))
  answerFile$X3 <-
    as.POSIXct(answerFile$X3, format = "%m/%d/%Y %H:%M:%S", tz = "UTC")
  user <- cert_r4_2_dataset[cert_r4_2_dataset$user == usernames[i], ]
  m <-
    match(answerFile$X3, user$date) #match answer file dates to user dates
  week <-
    user$week[m[1:length(m)]] #week in which the attack ACTUALLY occurred
##########Filter dataset to only include data relevant to chosen user. #######
  allWeeks <-
    split(cert_r4_2_dataset[cert_r4_2_dataset$user %in% usernames[i], ]$activity,
cert_r4_2_dataset[cert_r4_2_dataset$user %in% usernames[i], ]$week)
  indx <-
    sapply(allWeeks, length) #Convert the allWeeks variable into DataFrame.
  res <-
    as.data.frame(do.call(cbind, lapply(allWeeks, 'length<-', max(indx))))
#Reference:
http://stackoverflow.com/questions/15124590/column-binding-in-r
  ################################################
```

Algorithm 5: Continued.

```
  for (i in 6:length(res))
  {
    if (i <= length(res))
    {
      dl <- seq_dist(na.omit(res[i]), na.omit(res[i - 1]), method = "dl")
      jacc <-
        seq_dist(na.omit(res[i]), na.omit(res[i - 1]), method = "jaccard")
      cosine <-
        seq_dist(na.omit(res[i]), na.omit(res[i - 1]), method = "cosine")
          dlResults_temp <- c(dlResults_temp, dl)
          jacResults_temp <- c(jacResults_temp, jacc)
          cosResults_temp <- c(cosResults_temp, cosine)
    }
  }
  ##### DL ########
    highestDl = 0
    for (result in dlResults_temp)
    {
      if ((result) > highestDl)
      {
          highestDl = result
      }
    }
    dl_week = match(highestDl, dlResults_temp) + 5 #Offet is +5
since our results start at week 6.
    if (dl_week %in% week)
    {
      dlResults <- c(dlResults, dl_week)
    }
    else
    {
      dlResults <- c(dlResults, "FALSE")
    }
    ##### Jaccard ########
    highestJac = 0
    for (result in jacResults_temp)
    {
      if ((result) > highestJac)
      {
        highestJac = result
      }
    }
    jac_week = match(highestJac, jacResults_temp) + 5
    if (jac_week %in% week)
    {
      jacResults <- c(jacResults, jac_week)
    }
    else
    {
      jacResults <- c(jacResults, "FALSE")
    }
    ##### Cosine ########
    highestCos = 0
    for (result in cosResults_temp)
```

Algorithm 5: Continued.

```
        {
          if ((result) > highestCos)
          {
            highestCos = result
          }
        }
        cos_week = match(highestCos, cosResults_temp) + 5
        if (cos_week %in% week)
        {
          cosResults <- c(cosResults, cos_week)
        }
        else
        {
          cosResults <- c(cosResults, "FALSE")
        }
        ################
      }
    fullResults <-
        data.frame(usernames, scenarios, filenames, dlResults,
    jacRsults, cosRsults)
```

Algorithm 5: Distance measurement full evaluation code.

compute the distance for each individual technique in the time frame of minutes while the HMM technique took greater than 24 hours. As our data size of user activity increases (likely in real-life situations) thus so will the time taken to compute the datasets which may be a disadvantage in situations where computational resources are lacking.

The primary reason that distance measurements perform faster, in comparison with HMM, is due to the low complexity of such algorithms. In a distance measurement based approach, one simply needs to compare one set of data against a second set of data using the defined algorithm to calculate similarity (or dissimilarity). Thus, in the scope of this work, distance measurements will scale in a linear fashion relative to the number of weeks that need to be analysed for each individual user. HMM, on the other hand, requires one to consider the number of hidden states and starting parameters before performing a training phase to enable calculation of likelihood probabilities to take place. The training phase for the HMM is also required for each individual user and, in the case of using random restarts in the Baum-Welch algorithm (as performed by [12]), the time taken for each instance of training may be difficult to determine due to the random nature of this algorithm.

Based on the findings described thus far, it may be stated that no one technique is most suited in the detection of insider threats within the CERT r4.2 dataset. Although the HMM produces the highest detection ratio, none of the techniques have a 100% accuracy rating. Furthermore, given the low detection rate for each individual technique, no significant conclusion may be made on whether certain techniques are best suited for specific insider scenarios. The only exception to this statement is that HMM does show greater potential in the detection of Scenario 2 type activities. The results do demonstrate that there is potential in the use of distance measurements if we choose to combine them to come to a decision-making process (Table 4). By amalgamating the three distance techniques (i.e., if any of the three techniques match a specific week where insider threat occurs) we are capable of detecting up to 80% (56/70) of all insiders in the CERT r4.2 dataset (as opposed to 0.69 for HMM). Naturally, this amalgamation process may lead to a potentially greater number of false-positives if we apply it to benign datasets; thus future work is a consideration here.

7. Conclusions

7.1. Summary. This work has shown that distance measurements, including DL, Jaccard, and Cosine Distance, all have capability of the detection of insider threats within the CERT r4.2 dataset. In terms of detection rate, our results show that, out of 70 insiders, we are capable of producing a detection rate of 0.39, 0.36, and 0.47 for DL, Jaccard, and Cosine Distance, respectively. An aggregated score, through the combination of all three techniques, produces a total detection rate of 0.8. This is in comparison with 0.69 for the HMM technique as originally described by [12]. Our findings demonstrate that none of the techniques evaluated have capability to produce a 100% accuracy rate but distance measurements do have one noted advantage in place of more heavy weight machine learning algorithms: speed. While the HMM took greater than 24 hours to process all 70 malicious users' data, the distance measurements (both individual and combined) only took minutes to process. The advantage of speed may be of value in real-life scenarios whereby datasets of user activity may be significantly larger than the CERT dataset.

7.2. Limitations. In regard to real-life scenarios, we must acknowledge that there are certain limitations in the results presented in this, and most related works reviewed in this

```
library(HMM)
library(readr)
library(stringdist)
usernames <- c()
scenarios <- c()
filenames <- c()
hmmResults <- c()
files <-
  list.files(
    path = "~/answers",
    pattern = "*.csv",
    full.names = T,
    recursive = TRUE
  )
for (f in files)
{
  username <- regmatches(f, regexpr("-[A-Za-z0-9]+\\.", f))
  username <- sub("-", "", username)
  username <- sub("\\.", "", username)
  usernames <- c(usernames, username)
  scenario <- regmatches(f, regexpr("-[1-3]-", f))
  scenario <- sub("-", "", scenario)
  scenario <- sub("-", "", scenario)
  scenarios <- c(scenarios, scenario)
  filenames <- c(filenames, f)
 }
for (i in 1:length(usernames))
{
  answerFile <-
    read_csv(filenames[i],
             col_names = FALSE,
             col_types = cols_only(X3 = col_guess()))
  answerFile$X3 <-
    as.POSIXct(answerFile$X3, format = "%m/%d/%Y %H:%M:%S", tz = "UTC")
  user <-
    cert_r4_2_dataset[cert_r4_2_dataset$user == usernames[i], ]
  m <-
    match(answerFile$X3, user$date) #match answer file dates to user dates
  week <-
    user$week[m[1:length(m)]] #week in which the attack ACTUALLY occurred
  allWeeks <-
    split(cert_r4_2_dataset[cert_r4_2_dataset$user %in% usernames[i], ]$activity,
cert_r4_2_dataset[cert_r4_2_dataset$user %in% usernames[i], ]$week)
#Filter dataset to only include data relevent to chosen user.
  indx <-
    sapply(allWeeks, length) #Convert the allWeeks variable into DataFrame.
    res <-
      as.data.frame(do.call(cbind, lapply(allWeeks, 'length<-', max(indx))))
        ############### HMM Phase ########################
    hmm = initHMM(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), c(1, 2, 3, 4, 5, 6, 7))
#Initiate a 10 state HMM with 7 labels (which represent activities of user.)
  model = baumWelch(
    hmm,
    na.omit(unlist(res[1:5])),
    maxIterations = 20,
    pseudoCount = 0.1,
    delta = 0.01
  ) #Train our model with the first 5 weeks of user activity.
    vector = c()
```

Algorithm 6: Continued.

```
    for (i in 6:length(res))
      #For the remaining weeks of activity...
  {
      #What is the probability of a given observed sequence with respect to our model?
      logForwardProbabilities = forward(model$hmm, na.omit(unlist(res[i]))) #... calculate the
      probability of week i occurring against model...
      like <- ((logForwardProbabilities))
      lenthOfLike <- (length(like) / 10)
      answer <- sum(like[, lenthOfLike])
    vector[i - 5] <-
        answer #... store result of probability in vector...
    print(i) #Prints the current week to so we can see progress of computations in the console.
    model = baumWelch(
      model$hmm,
      na.omit(unlist(res[1:i])),
      maxIterations = 20,
      pseudoCount = 0.1,
      delta = 0.01
    ) #... and update model with week i.
  }
      ########## Find Lowest Probability #############
    probability = 0
    for (result in vector)
    {
      if ((result) < probability)
      {
          probability = result
      }
    }
    hmmWeek = match(probability, vector) + 5 #Find week which lowest probability occurred in. Offet is
    +5 since our results start at week 6.
    if (hmmWeek %in% week)
    {
        hmmResults <- c(hmmResults, hmmWeek)
    }
    else
    {
        hmmResults <- c(hmmResults, "FALSE")
    }
  }
fullResults <-
  data.frame(usernames, scenarios, filenames, hmmResults)
```

Algorithm 6: HMM full evaluation code.

paper since evaluation are conducted against synthetic datasets. As shown in the box plot of user activity of 'MCF0600' (Figure 4), there is significant uniformity in this users actions until the very week in which malicious activity occurs. We feel it is unlikely real-life activity would conform to such uniformity thus raising the question of how feasible all techniques described in this work would perform in real-life scenarios. Naturally, the ideal scenario would be to apply such techniques against real-life data sets captured within organizations but, due to confidentiality and privacy matters, this may not be ideal for researchers since comparison of results would be difficult if not impossible.

A second limitation related to distance measurements, from the implementation we have applied, is that only the prior week of a user's activity is compared against the current week (i.e., Week_{n+1} compared against Week_n). We note that this may have potential in producing false-positives in other datasets if a user's activity deviates greatly from one week to another (under the assumption that their activity is benign). Furthermore, the comparison of behaviour on week-by-week basis may produce false-negatives if an attacker stretches out their malicious behaviour in longer periods of time.

A third limitation of this work is that our evaluation is only conducted against known malicious users within the CERT 4.2 dataset. Since our current technique defines a malicious event based on the selection of the highest distance metric value, this approach would naturally raise a large amount of false-positives against benign users as well. The work of [12] solved this problem by applying thresholds to the probability values produced by their HMM and we believe

Table 5: Full evaluation results.

Username	Scenario	Answer filename	DL	Jaccard	Cosine	HMM
AAM0658	1	r4.2-1-AAM0658.csv	False	42	42	42
AJR0932	1	r4.2-1-AJR0932.csv	37	36	36	36
BDV0168	1	r4.2-1-BDV0168.csv	False	30	False	31
BIH0745	1	r4.2-1-BIH0745.csv	False	False	False	False
BLS0678	1	r4.2-1-BLS0678.csv	False	False	False	False
BTL0226	1	r4.2-1-BTL0226.csv	False	40	False	40
CAH0936	1	r4.2-1-CAH0936.csv	False	32	False	False
DCH0843	1	r4.2-1-DCH0843.csv	57	False	False	False
EHB0824	1	r4.2-1-EHB0824.csv	30	29	29	29
EHD0584	1	r4.2-1-EHD0584.csv	40	39	39	39
FMG0527	1	r4.2-1-FMG0527.csv	False	False	False	53
FTM0406	1	r4.2-1-FTM0406.csv	False	47	47	47
GHL0460	1	r4.2-1-GHL0460.csv	45	False	False	False
HJB0742	1	r4.2-1-HJB0742.csv	False	46	False	46
JMB0308	1	r4.2-1-JMB0308.csv	False	28	28	28
JRG0207	1	r4.2-1-JRG0207.csv	False	55	55	55
KLH0596	1	r4.2-1-KLH0596.csv	59	False	False	False
KPC0073	1	r4.2-1-KPC0073.csv	False	27	27	27
LJR0523	1	r4.2-1-LJR0523.csv	False	30	30	30
LQC0479	1	r4.2-1-LQC0479.csv	False	37	False	37
MAR0955	1	r4.2-1-MAR0955.csv	False	False	False	False
MAS0025	1	r4.2-1-MAS0025.csv	False	False	False	39
MCF0600	1	r4.2-1-MCF0600.csv	38	38	38	38
MYD0978	1	r4.2-1-MYD0978.csv	51	False	False	False
PPF0435	1	r4.2-1-PPF0435.csv	False	False	58	False
RAB0589	1	r4.2-1-RAB0589.csv	37	False	False	False
RGG0064	1	r4.2-1-RGG0064.csv	False	42	42	42
RKD0604	1	r4.2-1-RKD0604.csv	False	False	False	False
TAP0551	1	r4.2-1-TAP0551.csv	False	42	False	42
WDD0366	1	r4.2-1-WDD0366.csv	False	60	False	False
AAF0535	2	r4.2-2-AAF0535.csv	False	False	33	32
ABC0174	2	r4.2-2-ABC0174.csv	False	False	50	49
AKR0057	2	r4.2-2-AKR0057.csv	False	False	False	46
CCL0068	2	r4.2-2-CCL0068.csv	60	False	57	58
CEJ0109	2	r4.2-2-CEJ0109.csv	False	False	64	64
CQW0652	2	r4.2-2-CQW0652.csv	False	False	64	False
DIB0285	2	r4.2-2-DIB0285.csv	False	False	37	36
DRR0162	2	r4.2-2-DRR0162.csv	48	False	50	50
EDB0714	2	r4.2-2-EDB0714.csv	47	False	46	48
EGD0132	2	r4.2-2-EGD0132.csv	37	False	37	37
FSC0601	2	r4.2-2-FSC0601.csv	False	False	False	62
HBO0413	2	r4.2-2-HBO0413.csv	False	False	63	63
HXL0968	2	r4.2-2-HXL0968.csv	40	False	35	41
IJM0776	2	r4.2-2-IJM0776.csv	32	False	False	33
IKR0401	2	r4.2-2-IKR0401.csv	False	False	56	57
IUB0565	2	r4.2-2-IUB0565.csv	47	False	46	46
JJM0203	2	r4.2-2-JJM0203.csv	False	False	40	40
KRL0501	2	r4.2-2-KRL0501.csv	51	False	False	53

Table 5: Continued.

Username	Scenario	Answer filename	DL	Jaccard	Cosine	HMM
LCC0819	2	r4.2-2-LCC0819.csv	28	False	30	30
MDH0580	2	r4.2-2-MDH0580.csv	58	False	False	59
MOS0047	2	r4.2-2-MOS0047.csv	False	False	False	False
NWT0098	2	r4.2-2-NWT0098.csv	False	False	65	64
PNL0301	2	r4.2-2-PNL0301.csv	31	False	False	30
PSF0133	2	r4.2-2-PSF0133.csv	31	False	31	37
RAR0725	2	r4.2-2-RAR0725.csv	28	False	False	31
RHL0992	2	r4.2-2-RHL0992.csv	36	False	32	34
RMW0542	2	r4.2-2-RMW0542.csv	33	27	False	30
TNM0961	2	r4.2-2-TNM0961.csv	False	False	49	48
VSS0154	2	r4.2-2-VSS0154.csv	False	False	False	43
XHW0498	2	r4.2-2-XHW0498.csv	36	False	33	37
BBS0039	3	r4.2-3-BBS0039.csv	False	32	False	False
BSS0369	3	r4.2-3-BSS0369.csv	False	False	False	False
CCA0046	3	r4.2-3-CCA0046.csv	False	False	False	False
CSC0217	3	r4.2-3-CSC0217.csv	23	23	23	23
GTD0219	3	r4.2-3-GTD0219.csv	False	False	False	False
JGT0221	3	r4.2-3-JGT0221.csv	28	28	False	28
JLM0364	3	r4.2-3-JLM0364.csv	False	69	False	False
JTM0223	3	r4.2-3-JTM0223.csv	False	29	False	False
MPM0220	3	r4.2-3-MPM0220.csv	False	False	False	False
MSO0222	3	r4.2-3-MSO0222.csv	False	49	False	False

This table gives the full results of evaluation against all 70 insiders against each of the three distance measurement techniques described in this paper. An entry of "False" indicates that the technique failed to detect the correct week in which insider threat occurred while a numerical value denotes the week in which the highest measurement value was detected for an attack.

a similar technique may be applicable in the case of using distance measurements too. However, a thorough analysis of threshold techniques is outside the scope of this paper. Instead we provide a prototype of how thresholds may be implemented alongside other areas of future work in the final part of this paper.

7.3. Future Work. For future work, there may be value in experimenting with different time windows of analysis (i.e., calculating distance measurements of greater than or less than one week of user activity). We have focused on calculating the distance between individual weeks of user activity—there may be scope in analysing each individual users pattern of activity and reducing or increasing this window to produce greater detection rate. Secondly, we have focused specifically on analysis of users who are known to be malicious users and focused primarily on true-positives. Analysis of benign users and evaluation on the number of false-positives distance techniques produce would be of great value in this work. It is likely that optimisation of the distance measurement techniques demonstrated would be required in order to ensure a low level of false-positives is produced against benign users.

To reduce the number of false-positives, especially when applying the proposed techniques against benign users, some form of threshold will be required similar to the work of [12].

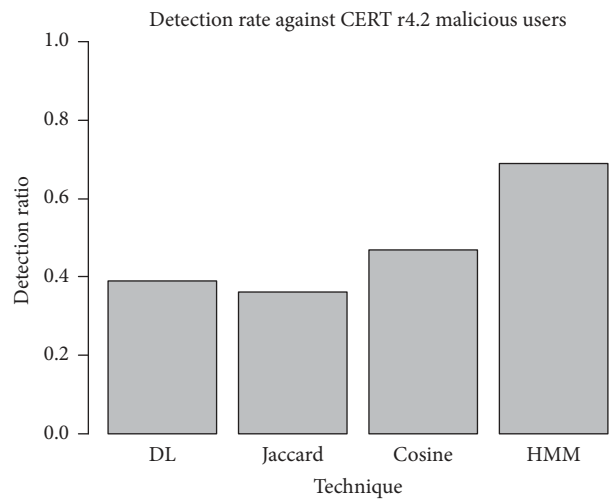


Figure 8: Detection ratio results.

An illustrative approach to applying thresholds is to use the z-score algorithm against the distance values produced for each user on a week-by-week basis. Figures 9 and 10 provide examples of a z-score 'smoothing' approach, originally written by [27], against user's 'MCF0600' and 'NGF0157' when the Damerau–Levenshtein distance technique is used. User

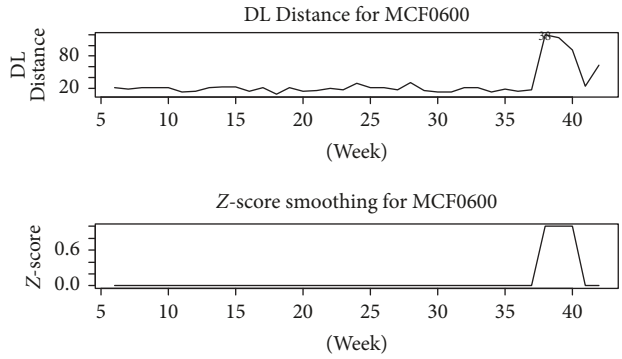


Figure 9: Applying Z-score smoothing to DL distance (malicious user MCF0600).

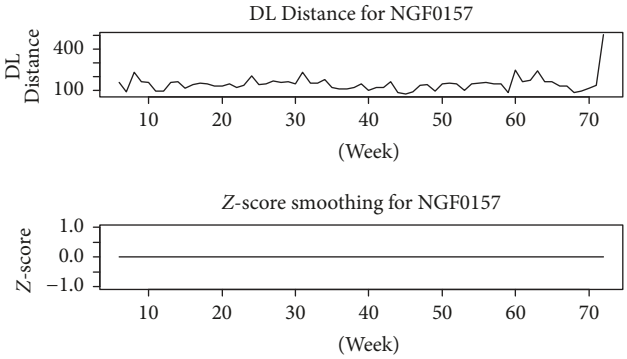


Figure 10: Applying Z-score smoothing to DL distance (benign User NGF0157).

'MCF0600' is considered a malicious user while 'NGF0157' is a benign user within the CERT r4.2 dataset. By applying the correct parameters (the parameters of lag = 30, threshold = 10, and influence = 0 were used) during the z-score smooth process, we can retain any peaks in our data which may be malicious behaviour while removing any peaks which are considered benign—therefore reducing the number of false-positives. Naturally, the choice of the correct parameters to use for each user will need to be accounted for; thus, we leave this task to future work.

Disclosure

Professor William J. Buchanan, OBE, Richard Macfarlane, Dr. Owen Lo, and Paul Griffiths are part of The Cyber Academy at Edinburgh Napier University, Edinburgh.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] E. Ukkonen, "Algorithms for approximate string matching," *Information and Control*, vol. 64, no. 1-3, pp. 100–118, 1985.

[2] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," *KDD Work. Data Clean. Object Consol*, vol. 3, pp. 73–78, 2003.

[3] A. H. Gray and J. D. Markel, "Distance Measures for Speech Processing," *IEEE Transactions on Signal Processing*, vol. 24, no. 5, pp. 380–391, 1976.

[4] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 116, pp. 374–388, 1976.

[5] A. Gionis, P. Indyk, and R. Motwani, "and others, Similarity search in high dimensions via hashing," *The VLDB Journal*, vol. 99, pp. 518–529, 1999.

[6] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: Min-Hash and tf-idf weighting," in *Proceedings of the 2008 19th British Machine Vision Conference, BMVC 2008*, UK, September 2008.

[7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[8] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, CRC Press, 2011.

[9] P. Nimbalkar, V. Mulwad, N. Puranik, A. Joshi, and T. Finin, "Semantic interpretation of structured log files," in *Proceedings of the 17th IEEE International Conference on Information Reuse and Integration, IRI 2016*, pp. 549–555, USA, July 2016.

[10] Z. Syed, A. Padia, T. Finin, L. Mathews, and A. Joshi, UCO: A Unified Cybersecurity Ontology,.

[11] C. Bizer, J. Lehmann, G. Kobilarov et al., "DBpedia—a crystallization point for the web of data," *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 154–165, 2009.

[12] T. Rashid, I. Agrafiotis, and J. R. C. Nurse, "A new take on detecting insider threats: exploring the use of hidden markov models," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats, MIST 2016*, pp. 47–56, Austria, 2016.

[13] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *AICS - Artif. Intell. Cyber Secur*, vol. 2012, 2017.

[14] B. Bose, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting insider threats using radish: a system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, vol. 11, no. 2, pp. 471–482, 2017.

[15] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, vol. PP, no. 99, 2015.

[16] A. Singh and S. Patel, "Applying modified K-nearest neighbor to detect insider threat in collaborative information systems," *Ijirset.Com*, vol. 3, no. 6, pp. 14146–14151, 2014.

[17] Y. Hashem, H. Takabi, M. Ghasemigol, and R. Dantu, "Inside the mind of the insider: towards insider threat detection using psychophysiological signals," *Journal of Internet Services and Information Security*, vol. 6, no. 1, pp. 20–36, 2016.

[18] B. Lindauer, J. Glasser, M. Rosen, and K. Wallnau, "Generating test data for insider threat detectors," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 5, no. 2, pp. 80–94, 2013.

[19] CERT, "Insider Threat Tools," https://www.cert.org/insider-threat/tools/, 2017.

[20] R Core Team, *R: A Language and Environment for Statistical Computing*, Austria, Vienna, 2016.

[21] J. Law, D. Mitarotonda, J. Larmarange, J. Boiser, and C. Hee, *Package lubridate*, 2016.

[22] A. Hadley, J. Hester, and R. Francois, Package 'readr' ,2017.

[23] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[24] L. Himmelmann, Package 'HMM', 2015.

[25] M. P. J. van der Loo, "The stringdist package for approximate string matching," *The R Journal*, vol. 6, no. 1, pp. 111–122, 2014.

[26] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," *International MultiConference Engineering Computer Science*, pp. 380–384, 2013.

[27] J.-P. van Brakel, "Peak signal detection in realtime timeseries data," 2017, https://stackoverflow.com/a/22640362.