Anomaly Detection Framework for Insider Threat

Muhamad Ibrahim
Dept of Computer Science & Software Engineering,
NED University of Engineering and Technology
email: *m.ibrahim2094@gmail.com*

Saad Kamran
Dept of Computer Science & Software Engineering,
NED University of Engineering and Technology
email: *saadkamran.sk@outlook.com*

***Astract* — This research presents a novel framework for detecting and reporting anomalous employee behaviour in an organisation, which would assist security analysts in responding quickly to insider threats. The study also compares four different deep learning architectures that comprise the framework's core detection algorithm. The designed system is an end-to-end architecture that performs data engineering, model application, anomaly flagging, and backtracking to source logs for additional information.**

Keywords — Insider Threat, Anomaly, Deep Learning, RNN-LSTM, cyber-security.

I. INTRODUCTION

Employees in an organisation have the most data access and knowledge of the security protocols of their company and their clients. According to the Computer Emergency Response Team (CERT) survey, employees or insiders pose greater threats to organisations than outsider or intruder attacks, accounting for 28% of all reported cybercrimes. Furthermore, according to the Breach Level Index, malicious insiders or accidental loss due to insiders are responsible for 40% of all data breaches. Ensuring that data is well protected and does not fall victim to a deliberate or unintentional breach is a colossal challenge for companies as they collect more and more data. Because of the nature of the data points, examining system log files is a complex task, and applying mitigation techniques by manually analysing the data is a time-consuming and taxing effort for security analysts. This comes at a high cost in terms of human resources and time constraints. When it comes to data breaches, time is of the essence because these breaches can grow in severity; often, the root cause of theft or loss is discovered days or weeks later.

This research focuses on addressing these issues in order to provide a streamlined anomaly detection framework that can be easily monitored for employee behaviour and allows for timely response by a security analyst. It is carried out on synthetic data obtained from CERT[1]. The data is specifically designed on insider information presented in the form of heterogeneous system log files of 4000 employees over a 1.5 year period. We also present a novel approach to engineering this data for use as input to our deep learning models, as well as a method to visualise the output of subsequent results. Nonetheless, we have provided a mechanism to cross-check an employee's activity and confirm malicious practise prior to confronting the suspect, making our entire system more powerful and robust.

II. LITERATURE REVIEW

In recent years, there has been an increase in research on the use of artificial intelligence and various machine learning and deep learning models in the cyber-security space, with a focus on detecting insider threats. While study and experimentation for accuracy and precision of different modeling architectures have been prevalent in research, less attention was given to designing an end-to-end framework that would allow for better analysis, detection, and swift decision making, all of which is crucial in designing an operational cyber security system.

Substantial amount of research work also focused on utilizing descriptive modelling techiniques that would try to model on events that have had already occurred. The disadvantage of this approach is that it would take time to model on large amounts of data, allowing enough time for the attack to intensify or the prepatrator to leave the organisation. Contrary to this our approach is based on methodology of predictive modelling. Based on historic trends of system usage, which we consider as normal behavior, users' probable activity scores are predicted for the next week. When actual events are logged, the real scores are compared with the predicted scores of the model and errors are calculated in near real time, allowing for continous threat detection.

III. SYSTEM DESCRIPTION

A. System Overview

Figure 1 depicts the architectural model that defines the basic components that work together to form the entire system. The architecture is based on the input from the event log files

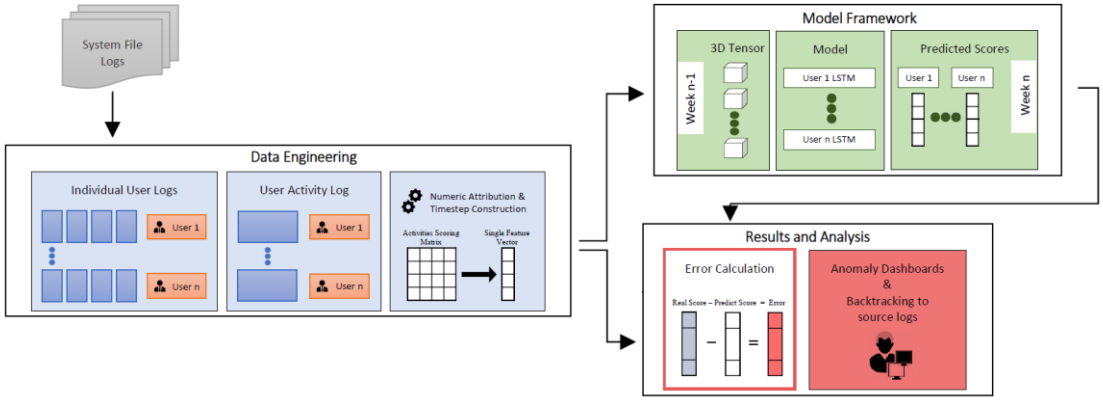

Figure 2: System Architecture

and routes it through various stages. The goal is to predict abnormal patterns in user behavior that can be flagged; it will then be the primary responsibility of a security analyst to determine whether this deviation was malicious in nature. To accomplish this, all system logs are traversed, and each user's profile is extracted, as well as the users' behaviors, which are numerically attributed and fed into the deep learning model. Because it is critical that data threats are detected as soon as possible, we chose to train on historic data users and output predictive scores of user normal behavior for the following week rather than opting for unsupervised descriptive modelling, which would try to model on user's actual data to try and flag anomalies. These scores are saved and compared as new data from system logs is received, allowing for the calculation of errors (predicted scores vs actual scores) and the detection of anomalies on threshold values. These anomalies are visible as abnormal spikes in error trends for each activity type (for example, email, web, file download/upload, and so on). The system then identifies these threat events and allows security analysts to trace them back to the original source logs to determine whether the activity was truly malicious or merely abnormal.

B. Data Collection and Preprocessing

As previously discussed, the data was gathered from CERT, which consists of system log files of employee activities of various types. The ones we chose for our research were taken as follows:

System Type	**Attributes**
Logon	id, date, user, PC, activity of Logon/Logoff
Device	id, date, user, PC, file tree, activity of Connect/Disconnect
Http	id, data, user, PC, URL, activity of Visit/ Download/Upload ,content.
Email	id, date, user, PC, to, cc, bcc, from, activity of Send/Receive/View, size,attachment, content
File	id, date, user, PC, filename, activity of Open/Copy/Write/Delete, to removable device and fromremovable device, content.

After data is loaded from these file systems it is combined to form one view that has all types of activities against a user. Following that, is to numerically attribute each user activity making it suitable for input to our model. This is achieved through a refined mechanism adopted from [2].

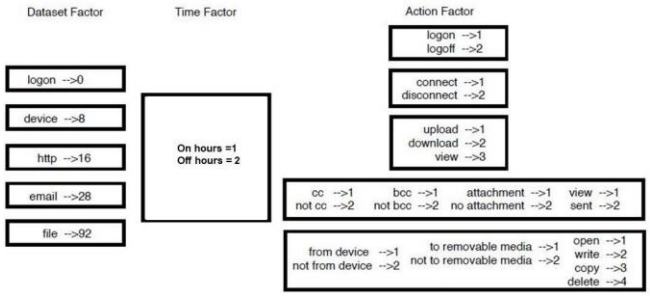

Figure 1: Numeric attribution

It is important to note that columns used only in modelling are kept moving forward and rest are dropped. The dropped columns comprise of the ones we cannot intuitively assign a numeric representation to, for example content and URLs. Our data is organized as a time-series pattern, and we simplified our trends by keeping only working days and removing weekends and employee holidays. To account for erratic trends, we imputed hourly activity for each hour, the dataset factor and all others were set to 0 if no activity occurred, but a score was assigned to each hour, resulting in a more streamlined sequence moving forward to the next step. On/Off work hours may differ for each employee or department under consideration, so this must be divided accordingly. NA values are assumed to be 0 by the model, which treats them as nulls. When all activities are numerically attributed final scores are calculated with the following formula:

*Dataset Factor * [(sum of activities) * on/off]*

By this we get scorings distinctive in range for each log type. These scores are referred to as real scores in rest of the paper. After scoring our activities, we then roll up and aggregate the

scores on a novel timestep sequence: "Date_On/Off Hour_Type". This gave 10 unique timesteps for each day as shown in Figure 3.

Final Rolledup Scoring - DataFrame

Index	timetype	score
240	2010-02-05_1_0	3
241	2010-02-05_1_8	0
242	2010-02-05_1_16	3068
243	2010-02-05_1_28	526
244	2010-02-05_1_92	288
245	2010-02-05_2_0	0
246	2010-02-05_2_8	0
247	2010-02-05_2_16	0
248	2010-02-05_2_28	0
249	2010-02-05_2_92	0
250	2010-02-08_1_0	4
251	2010-02-08_1_8	0
252	2010-02-08_1_16	3078
253	2010-02-08_1_28	530
254	2010-02-08_1_92	0
255	2010-02-08_2_0	0
256	2010-02-08_2_8	0
257	2010-02-08_2_16	0
258	2010-02-08_2_28	0
259	2010-02-08_2_92	0

Figure 3: Rolled up timestep scoring

Our deep learning model requires a 3D tensor input. After careful considerations, the most optimal value of timesteps chosen to be 50, which essentially represents one-week.

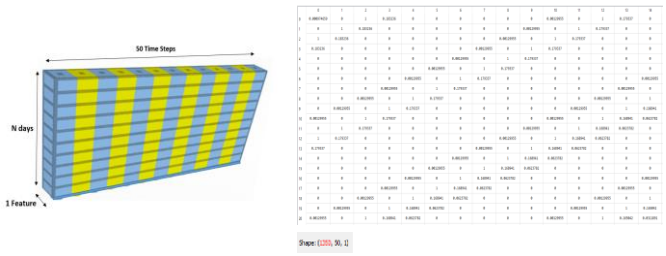

Figure 4: 3D tensor as final input to Mode

Note that we have one single feature that is the timestep score. This structure takes 50 timesteps as input and the next timestep becomes its output that it needs to learn.

C. Modelling Techniques

The research concluded on experimenting with 4 different architectures of LSTMs. LSTM units have been designed to enable Recurrent Neural Network (RNNs) to tackle problems that require long-term memories. LSTM units consist of cells of memories that gather and accumulate information, which are further connected to the next time step. The new values are augmented in the memory cell by producing the new input and a forget gate is assigned that weights the newer and older information. The architectures studied were namely, Unidirectional LSTM, Bidirectional LSTM, Auto-Encoder LSTM and CNN-LSTM.

In Unidirectional LSTM, the information is processed from the inputs that have already been parsed through it using the hidden state. In this variation, past information is preserved because the only inputs it has seen are from the past. As an advancement to this, the introduction of Bidirectional LSTM has utilized two LSTMs in training the input – one from past to future and one from future to past, allowing it to learn more information. An auto-encoder LSTM implements the data using encoder-decoder architecture and the evaluation of the performance is done on the ability to recreate the input sequence. CNN LSTM apart from spatial, also allows temporal sequence inputs that can be leveraged in time-series modelling. It uses a convolutional layer on top of the LSTM structure giving an additional ability to perform feature extraction. Models are trained on an 80-20% split and since user behavior in real world cases can be evolving system allows for training in an online fashion either daily or weekly to keep up with the dynamic trends of the user.

D. Reporting and Data Visualization

Our anomaly detection is based on error calculation, which is defined as deviation from normal usage patterns. Predicted scores generated from week 'n-1' data, and real scores of week 'n', are differentiated on each timestep and squared: to return absolute value and place emphasis on larger deviations. Error scores for each log type are separated and scaled from 1-100. The first threshold is applied on error scores to capture above 80^{th} percentile for each log type. This is then further combined into a single dataset giving top errors respectively on each log type in a similar range since this had been scaled earlier in the previous step. This new dataset is now rolled up on timesteps and error sums are taken. The second threshold is applied on timesteps at an optimal value of 15, showing highest deviating 15 timesteps in the dashboard. This would cover 90-95% of anomalous activities as experimented with our test cases.

The insights dashboard provides information to the security analyst that will aid him/her to identify potential malicious practice by visualizing top deviations and trend over time. This is achieved with the following two types of dashboard consolidation. Figure 5 illustrates a timestep consolidation over previously set 15 plots, where normal behavior can be seen in grey and deviating activity with red across each system log type. In figure 6, the type consolidation view shows trend for each log type over time with red flags marked as anomalous events.

Viewing these dashboards with narrowed down results and highlighted anomalies, security analysts get a good estimate where to look for and what to look for. The backtracking subsystem allows to pinpoint to exact activities that are suspected to be malicious. The security analyst can efficiently traverse through actual system files to look at detailed information of these events such as content shared, and activity

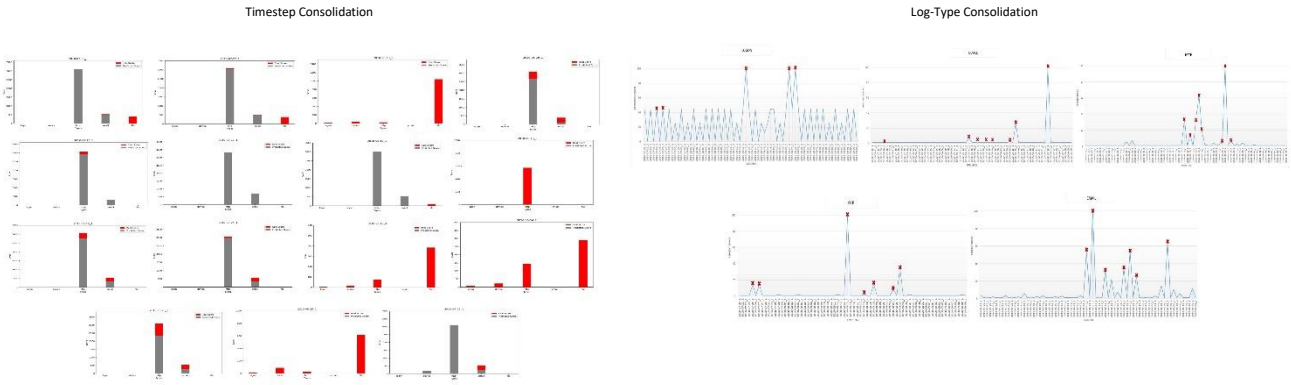

Figure 5: System Dashboards

performed. In the end, it is always the job of the security analyst to assess whether a deviation can be truly an insider breach or a false alarm. This part gives room to get confirmation on exactly what happened before an employee is called in or issue is escalated for further investigation.

IV. EXPERIMENTATION

A. Dataset, Test Cases and Evaluation

As previously highlighted the data was acquired from CERT Insider Threat dataset r6.2. It has data for 4000 employees for a period of 18 months. The dataset also provided test cases for malicious users and detailed a description of their activities. Our proposed system was evaluated on two of these test cases.

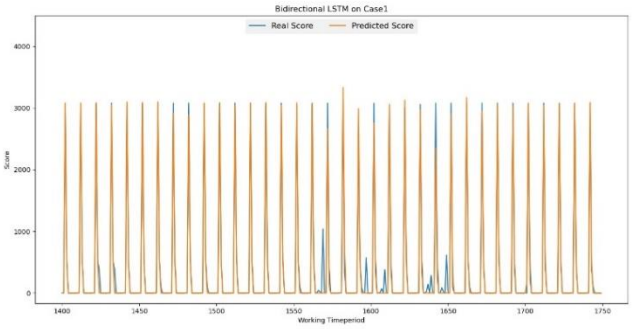

Figure 6: User Case 1

The first user ACM2278, who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to the internet. The user then leaves the organization shortly after. The second test case, user CDE1846, logs onto another user's machine and searches for interesting files, emailing to their home email. This behavior occurs more and more frequently over a 3-month period. In both figures 6 and 7, we compare real scores coming in real time with the models predicted scores from week n-1. We can see little spikes of anomalous activity that showed out of trend for what the model predicts. When we take the difference of these scores, these are returned as errors.

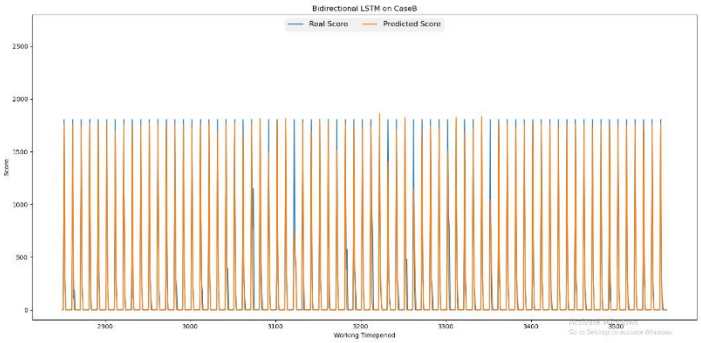

Figure 7: User Case 2

B. Results

Our results showing performance of our models were measured in three metrics: Precision, Recall and f1-Score. Precision is measurement of correctly predicting our positive outcomes from all outcomes marked as positive; in our scenario it can be taken as flagging a non-anomaly as an anomaly. The relevancy of this is to gauge the extent to which our system produces false alarms, a high precision ensures that false alarms are low, avoiding the system to be futile. Recall is measurement of predicting our positive outcomes from all actual positive cases; it places an emphasis on our false negatives, in this case flagging anomalous behavior as anomalous. It is of much more importance since missing out an insider activity can result in severe damage to the organization and our systems actual purpose would be not achieved. Instead of accuracy we measure the f1-Score to gauge the best model since both our metrics of precision and recall are indispensable, so f1-Score allows us a balance between these two metrics and also helps with the uneven class distribution. Table I shows results received for both our user cases. We can see that bidirectional LSTM architecture due to its superior capabilities of learning trends in forward and reverse temporal directions outperforms other models in overall metrics. Unidirectional LSTM variant showed more stable performance for Precision, but since Recall is of more importance in cyber-security threat detection, it became the deciding factor. CNN-LSTM and AutoEncoder models for case1 wasn't able to fit on the data and could not

TABLE I
INSTANCE-BASED TEST RESULTS

LSTM Architecture	User-ACM2278			User-CDE1846		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Unidirectional	78.75%	73.88%	76.15%	83.82%	85.07%	84.44%
Bidirectional	**39.22%**	**90.91%**	**54.8%**	**85.91%**	**95.52%**	**90.46%**
AutoEncoder	NA	NA	NA	37.5%	20.15%	26.21%
CNN	NA	NA	NA	63.32%	75.37%	68.70%

even reproduce the training data. One reason could be because of the parameters of dropout layers we set to avoid overfitting resulted in the model to generalize more than memorize the training data. It did however produce comparable results for case 2, where we get some level of predictive power from them.

V. CONCLUSION AND FUTURE WORK

In this paper, we address Insider threats, which is a growing challenge for organizations aiming to maintain privacy and security of their own data and that of their customers or clients. We present an applied AI end to end solution for anomaly detection of such insider activity, that can aid security analysts in taking swift decisions in near real time. Our solution leverages deep learning LSTM models at the core of our detection algorithm that evaluates employee behavior along different sets of system logs and outputs any high errors or anomalies from the expected behavioral trend. The LSTM architecture which best works for our system is the Bi-directional LSTM giving commendable Recall with decent Precision scores. This is closely followed by the uni-directional LSTM architecture which was able to perform fairly decently as well with robust Precision. AutoEncoder and CNN-LSTM variants did not perform well and results did not look promising.

In future work, we will look towards experimenting this system on other datasets and try to incorporate information captured in content variables through further exploration of NLP algorithms.

VI. ACKNOWLEDGEMENT

This research was conducted as part of the final year project for the undergraduate program in computer science offered by NED University of Engineering and Technology's Computer Science and Software Engineering Department.

VII. REFERENCES

[1] "CERT Insider Threat data set library" https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099

[2] "Insider Threat Detection with Long Short – Term Memory" by Jiuming Lu and Raymond K. Wong, ACSW, 2019.

[3] "Deep Learning for Unsupervised Insider Threat Detection in Structured Cyber security Data Streams" by Aaron Tuor, Samuel Kaplan, Brain Hutchinson, Artificial Intelligence, 2017.

[4] "A New Take on Detecting Insider Threats: Exploring the use of Hidden Markov Models" by Tabish Rashid, Ioannis Agrafiotis, Jason Nurse, ACM, 2016.

[5] "Supervised Learning for Insider Threat Detection Usig Stream Mining" by Pallabi Parveen, Zahary Weger, Kevin Hamlen, Latifur Khan, Air Force Office, 2012.

[6] "A Survey of Deep Learning Methods for Cyber Security" by Daniel Berman, Anna Buczak, Jeffrey Chavis, and Cheritta Corbett, MDPI, 2019.

[7] "Insider Threat Assessment: a Model – Based Methodology" by Nicola Nostro, Francesco Brancati, DISCCO, 2013.

[8] "Classification of Insider Threat Detection Techniques" by Ameya Sanzgiri and Dipankar Dasgupta, Rights Link, 2016.

[9] "Toward an Ontology for Insider Threat Research: Varieties of Insider Threat Definitions" by David Mundie, Sam Perl, Carly Huth, IEEE, 2013.

[10] "Combating Insider Threats by User Profiling from Activity Logging Data" by Muhammad Dahmane and Samuel Foucher, ICDIS, 2018.

[11] "A Trust Aware Unsupervised Learning Approach for Insider Threat Detection" by Maryam Aldairi, Leila Karmini, and James Joshi, IEEE, 2019.

[12] "Deep Learning Based Attribute Classification Insider Threat Detection for Data Security" by Fezhni Meng, Fang Lou, Zhihing Tian, and Yungshen Fu, IEEE, 2018.

[13] "Machine Learning based Insider Threat Modelling and Detection" by Duc Lee and Nur Zincir Heywood, IEEE, 2019.