ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 7, July 2023

Using Restricted Boltzmann Machine for the Detection of Insider Attack in Machine Learning

[1] Prof. Nandini K*, [2] Dr. Girisha G S, [3] Adarsha Subrahmanya K K, [4] Bharath Kumar N, [5] B Bhargav Ram C S, [6] Chethan R

[1] [2] [3] [4] [5] [6] Department of Computer Science and Engineering, Dayananda Sagar University, Bangalore, Karnataka, India
Corresponding Author Email: [1] nandini-cse@dsu.edu.in, [2] girisha-cse@dsu.edu.in,
[3] adarsh930408@gmail.com, [4] bharathkumarbharath075@gmail.com, [5] ramcsbhargav505@gmail.com,
[6] chethanrraj90@gmail.com

Abstract— ***The analysis of a company's computer network activity is crucial to early detection and diminution of insider threats, which are of growing concern to many businesses. These extreme harm acts, such intellectual property theft and the publication of sensitive information, are typically carried out by authorised users. An online unsupervised learning in deep learning proceeds towards detect abnormal network activity from logs of system is presented as a prospective filter for human analysts. Restricted Boltzmann Machines (RBMs) have been the subject of some exploratory research as a network intrusion detection strategy. The key difficulty with current technological progress is to recognize the internal danger within the cloud network. Once information is lost, cloud users are harder to compromise. When security and confidentiality aren't guaranteed, cloud computing is unreliable. The likelihood that sensitive applications like banks, hospitals, and companies may be harmed by genuine user threats is higher. An invader is described as a network user and is displayed as a user. After entering into the network as an insider, they will try to attack important information as it is being communicated or exchanged. The network of a cloud has a number of options for external security. The way of identifying an inside assault utilizing AI technology is the main topic of this study. Utilizing nodes of vulnerable user systems makes it feasible for an inside assault. They will connect to the network using a poor user ID, log in, and make the assertion that they are a trusted node. Then, it is quite challenging to recognize them and they may easily hack and assault information as insiders. In our proposed study, we keep an eye on attackers using a deep learning method and a model of user interaction behaviour. User activities from the actual user are recorded in a database. Furthermore, it will notify the system of an insider threat. Analyse the behaviour of individuals within the organization that is malicious. Our main intention from this project is to provide user-friendly software applications. The system detects insider threats, alerts the system, and identifies malicious behaviour within the organization.***

Keywords—Restricted Boltzmann Machines (RBMs), Artificial intelligence, Security, Insider Attack, User Interaction Behaviour, Deep Belief Neural Network.

I. INTRODUCTION

The majority of security vulnerabilities inside the cloud network may be resolved using Machine Learning (ML) as well as Deep Learning (DL) approaches. To find a better solution, one of them is strongly driven to identify user behavior in numerous aspects. Machine learning models facilitate the classification of observed information through knowledge of the characteristics or properties learned from training data. There are two kinds of ML, Supervised learning and Unsupervised machine learning.

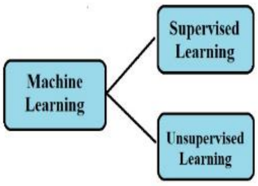


Fig. 1. Classification of Machine Learning

Supervised learning, as its name suggests, includes the presence of a supervisor as an instructor. Supervised learning involves teaching or training the machine with well-labelled data. After receiving a fresh batch of examples, the computer uses the supervised learning algorithm to analyse the training dataset and generates the right answer using labelled data. They are the models that need labels based on ground truth for training. The intention or aim is to predict whether the next attempt of the user will fail. Unsupervised machine learning differs from supervised models in that it does not call for a teacher or supervisor, which implies the machine will not be trained. In this case, the machine's main objective is to classify unsorted data based on patterns, similarities, and differences even without previous data training. As it does not need labels for training.

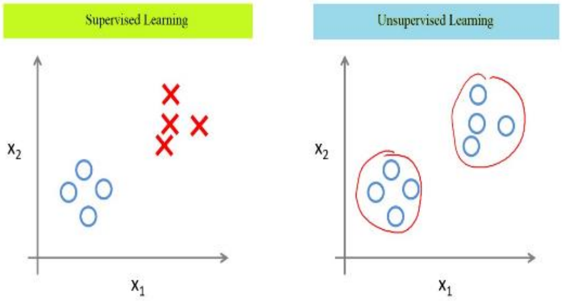


Fig. 2. Classification of Supervised Learning and Unsupervised Learning

A company insider is someone who works for or has worked for the organisation and still has control/access to the information of the technological architecture. Insider comprises of 2 types they are malicious and non-malicious insiders.

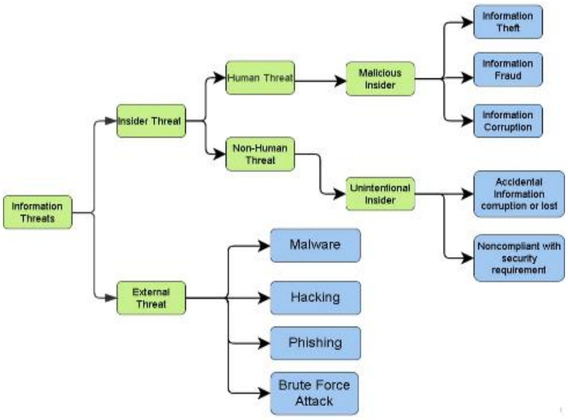


Fig. 3. Classification of Malicious insider and Non-Malicious Insider

A malicious insider is a person who has control to sensitive information within an organisation and deliberately utilises it against the interests of the company. This individual could be a partner in company, a contractor, an employee (current or past). On the other hand, a non-malicious insider poses an inadvertent threat as a result of carelessness or neglect in the performance of a typical day-to-day duty. One of the most significant undiscovered dangers to protected data has been found as being this. An insider threat is a former or current employee, contract worker, or business associate who purposefully compromises the integrity, confidentiality or accessibility of the organization's data or information systems and has or had access privileges to the network, system, or data. Insider attacks are destructive behaviours committed by an organisation member with the necessary power. Threats made by hostile persons, such as being granted access to the business's network, systems, and data, have been used to harm the secrecy of the organisation. For intranets that must meet security standards, internal assaults have long been one of the most significant problems since they might endanger the entire system.

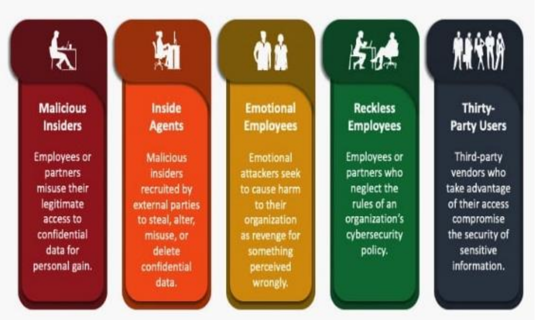

Malicious Insiders
Employees or partners misuse their legitimate access to confidential data for personal gain.
Inside Agents
Malicious insiders recruited by external parties to steal, alter, misuse, or delete confidential data.
Emotional Employees
Emotional attackers seek to cause harm to their organization as revenge for something perceived wrongly.
Reckless Employees
Employees or partners who neglect the rules of an organization's cybersecurity policy.
Thirty-Party Users
Third-party vendors who take advantage of their access compromise the security of sensitive information.

Fig 4. Classification of Different Types of Employees Inside a Company

In recent years, the security of the intranet has received increasing attention due to the frequency of internal threats. In order to identify intruder who are likely to be able to control the cloud information, the DBN (Deep Belief Neural Network) is used as a classifier. The characteristics of the insider are derived from their behavioural engagement with the programme through the user logs, such as Logon/Logoff activity, File, HTTP (HyperText Transfer Protocol), Email, Device, Keystroke & Mouse activity. These user logs' aberrant access is computed and utilised as an accent for the categorization of insiders. The majority of security challenges in the network of cloud can now be solved using ML and deep learning approaches. The Restricted Boltzmann Machine (RBM) is used in the construction of the deep belief neural network to enable communication between the RBM layer and layers above and below it [5]. Using the insider detection approach, it is possible to identify the unreliable or malicious nodes inside the organisation. This method offers protection from harm and detection before an assault. Our main objective at the end of the project is to provide users with an application malicious behaviour within the organization, and identifies the behaviour.

II. RELATED WORK

According to our literature review, it is noted that a complete application of ML approach to the detection of insider attacks is being made in every system. Comparing ML approaches to modern prediction methodologies, ML techniques are proven to be significantly quicker and with more accuracy. In this area, a lot of work has been done worldwide. The writing of several authors serves as proof of this,

Tamer Aldwairi produced the concept of **"An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection"** at 2018. This paper conveys the attack which took place in computer network and they discuss about technique called **A-NIDS** which means another anomaly network instruction detection system which helps to detect the new pattern of the attack which did not happened even before and they used the machine learning techniques such as RBM which helps to distinguish between the normal and the anomalous patterns. So, here we understand how an attack will happen running in a computer network and we get the insights about RBM which will be used for separating normal patterns and the anomalous patterns [5].

The paper produced the concept of **"Review on insider threat detection techniques"** in 2019. In this paper basically an employee is the attacker. When he becomes against the company and misuse the information in an unethical way. So, how we can find him by some of the techniques. Here we usually use machine learning and non-machine learning approaches. Non machine learning methods are **access control**, blockchain technology. And other techniques are machine learning such as **K Nearest Neighbor (KNN)**, **decision tree**, **back propagation neural technique**, **naïve bayes**. So, by using these techniques which helps to give better accuracy to find an insider in a company [6].

The paper produced the concept of **"Insider Attack Detection Using Deep Belief Neural Network in Cloud Computing"** in 2021. This paper is about the insider attacks which will take place in the cloud computing domain. Where the organisation will keep sensitive information on the cloud. When this information is exposed, attackers may abuse it. So, they used techniques like the hidden **Markov dynamic method** to detect the attack in the cloud and other techniques like **long short-term memory** and **support vector machine**. And from this study, we learned or referenced to knowledge on how an attack occurred in the cloud, how to employ a deep belief neural network, and how to manage data in the cloud [7].

The paper illustrated the concept of **"Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams' in** 2017. In this paper attacking will take place under the cyber security domain to prevent the possible insider threads using data streams. Basically, they have used **deep learning** concepts for the unsupervised which means unlabelled insider threat data for analysis of the computer network actively and for detecting the attack. So, what we learn from this paper is how to use the information about insider attacks and how to manage them and some of deep learning concept in the security domain and along with that how to manage those data [8].

Another paper presented the concept of **"Detecting insider threat within institutions using CERT"** CERT (Computer Emergency Response Team) data sets which is the insider data set and apply the machine learning methods such as **random forest**, **Naïve bayes**, **K nearest neighbour dataset and different ML techniques' in** 2021. They use **algorithm** to measure the accuracy and the error data rate. Our understanding of the CERT dataset and how to utilize it in our work to train machine learning models and assess their correctness is what we have been referring to in this study. From there, we may use those tools for future data and user resources to anticipate the insider danger in cyberspace [9].

The Paper [10] presented the concept of "An Insider Threat Detection Approach Based on Mouse Dynamics and Deep Learning" in 2019. In this paper they focused on the mouse activity and they will collect the data of the mouse of the user and they will pre-process the data into unnecessary once and necessary once. They have used the method Dynamic mapping which helps in distribution of those data. The author used the CNN (Convolutional Neural Network) network of deep learning to automatically extract and model the user behaviour photos by mapping the user's mouse actions into images. So, what we infer from this paper is how to handle mouse data and how to use mouse data to find insider attacks which helps to increase accuracy of the model [10].

III. METHODOLOGY

Because cloud security is such a big problem, attackers may steal important data from the cloud's data sources. Even though the cloud network is already safe, there are still internal attackers. In this case, internal or insider attackers gain access to crucial system data. The loss of data is caused by this information infiltrating from server of cloud through an insider, which is a serious problem. In the research study, ML approach are employed frequently for cloud security. In this study, the combination of insider interaction behaviours, including keystroke, mouse dynamics, logon/logoff activity, file activity, HTTP activity, email activity, and device activities, are taken into account and used for feature extraction with a deep learning algorithm called deep belief network in order to detect insider threats (DBN). In the cloud network, DBN is utilized to forecast insiders' unauthorized behaviour. DBN are feed forward neural networks with a deep architecture and numerous hidden layers that are machine learning algorithms that mimic deep neural networks but are not the same. Unsupervised, straightforward networks like RBMs (restricted Boltzmann machines). A Deep Belief Network will be created by connecting these restricted Boltzmann machines in a certain sequence. The outcome of the Boltzmann machine's "output" layer is continually fed into the subsequent Boltzmann machine as input. Then, we'll train till it converges and continue using the same strategies until the entire network is built. A network of unpredictable processing units linked in both directions is referred to as a Boltzmann machine, or BM. In this case, a BM's nodes may be classed as visible or concealed nodes. Nodes that are visible signify a part of an observation. For instance, each and every pixel in a computer, each visible node, digital picture, in an image categorization. However, dependencies among visible nodes that are not modelled by simple pairwise interactions across visible vertices are captured by hidden nodes.

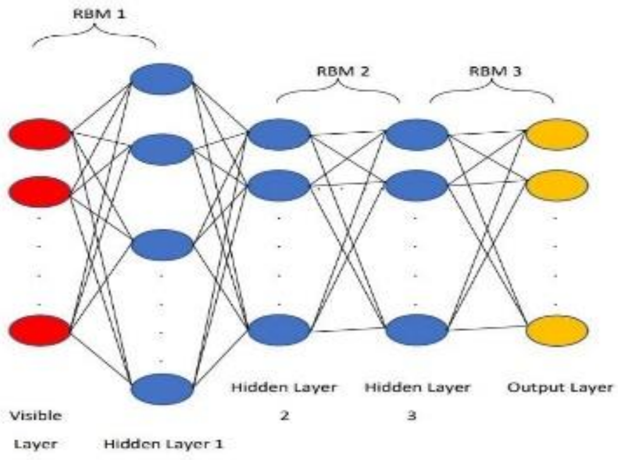


Fig. 5. DBN with RBM

The Deep Belief Network (DBN) is created by arranging RBMs in a stacked arrangement and then training them with a specific learning rule. In the training process, synaptic weights between different layers, biases, and the states of neurons are taken into account. The transformation of bias and neuron state from one layer to the next is done with a sigmoid function, as expressed in

$$\Phi(r_i 1) = \frac{1}{1 + \exp\left(-a_i - \sum_j r_j y_{ij}\right)} \qquad (6)$$

The postsynaptic bias and weight of the neurons in the RBM layer are initialized at the beginning. There are two steps for every piece of input training data: positive and negative. Data is transformed from the input nodes to the middle layer during the positive stage and back again during the negative stage from the hidden layer to the visible layer. Equations 7 and 8 are used to calculate the activation of each phase, respectively.

$$\Phi\,(v_i = 1|\kappa) = \sigma\iota\gamma\mu\left(-a_i - \sum_j k_j\, y_{ij}\right) \qquad (7)$$

$$\Phi\,(k_i = 1|\mu) = \sigma\iota\gamma\mu\left(-d_i - \sum_j k_j\, y_{ij}\right) \qquad (8)$$

The weight values are manipulated till the maximal number of epochs is reached, as opposed to the regular DBN. As training continues, the settings are optimized by

$$\Upsilon\pi\delta\alpha\tau\varepsilon\left(y_{ij}\ \frac{n}{2}\, X\ \left(positive(E_{ij})\ negative(E_{ij})\right)\right) \qquad (9)$$

Here the positive(E_{ij})-Positive statistics of edge

$E_{ij} = f(k_j = 1|\mathrm{m})$, negative$(E_{ij})$-Positive statistics of the edge $E_{ij} = f(m_j{=}1|\mathrm{k})$, n - learning rate.

Boltzmann machine generally referred to as BM, each node in a network with bidirectional connections is classified as being either viewable or hidden. A BM with m visible nodes may be seen here.

A. Feature Extraction of user behavioural characteristics

Security threats brought on by personnel within the company follow insider threats. These unauthorized accesses to the information might have a detrimental impact on the organization's policy and result in data loss. Assailant employees and civilized employees are the two categories into which these workers fall. In this study, the attacker i from the company is discovered by observing their interaction behaviour that they are functioning with their appropriate jobs, but their behaviours are not normal. Therefore, the regular users are classified as attackers or malevolent users depending on their working behaviour. All the user activities are the categories used to describe how users interact with computers. Therefore, a user may communicate with any types of data as well as programs in the cloud system using these kinds of operations [7].

B. Approach of CERT Dataset

Getting access to genuine business system logs is quite challenging. The "CERT Insider Threat Tools" dataset was therefore employed. In this case, the CERT dataset is an intentionally constructed dataset used to test insider-threat detection algorithms rather than genuine business data. Employee system usage user activities are included in this dataset, along with certain organizational data like employee departments and responsibilities. Each table has columns for a user's ID, timestamps, and actions, such as the kinds of usage data, the quantity of variables, the number of workers, and the quantity of harmful insider activities, which vary based on the dataset version.

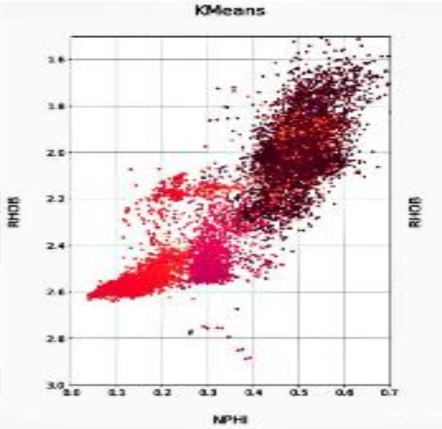


Fig. 6. Above Cross plot Showing in the Different Unsupervised Learning Clustering Methods.

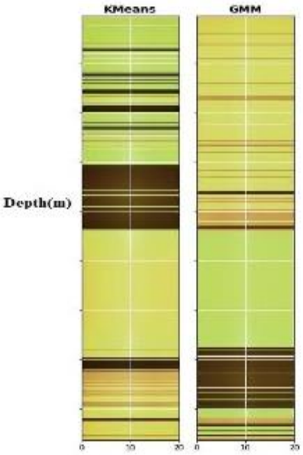


Fig. 7. Above Log Plot Showing in the Different Unsupervised Learning Clustering Methods.

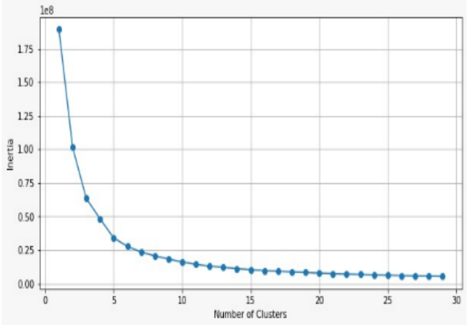
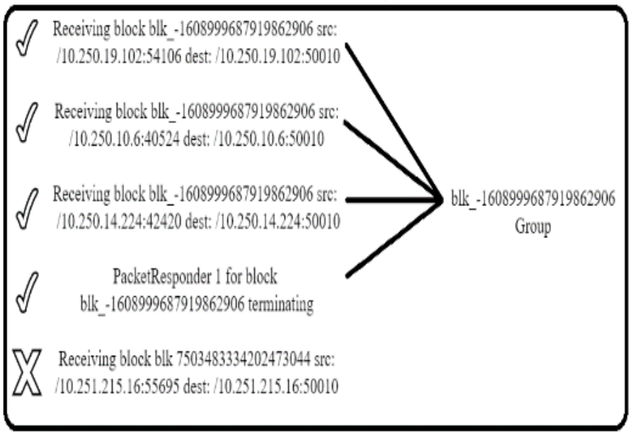


Fig. 8. Above Elbow plot Showing Optimum Number of Clusters vs Inertia.

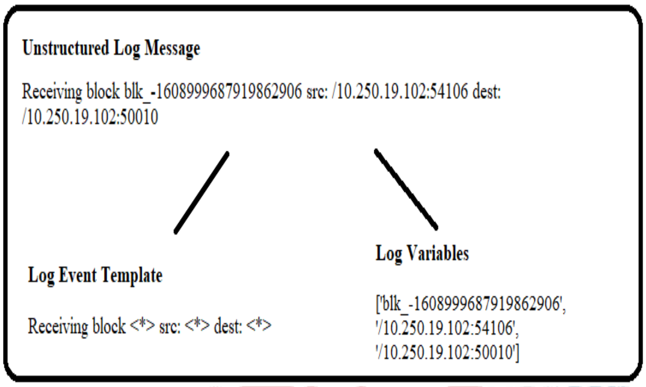
C. Approach of HDFS dataset

The Hadoop Distributed File System, or HDFS, was created to function on standard hardware. Due of HDFS's widespread use, a lot of research has been done on it recently. The benchmark in a cloud - based system, workload, and manual labeling using specially created criteria were used to create this log collection. Block ids are used to divide the logs into traces. A specific block id is then given each trace connected to a ground truth label.



```
✓ Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010
✓ Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010
✓ Receiving block blk_-1608999687919862906 src: /10.250.14.224:42420 dest: /10.250.14.224:50010
✓ PacketResponder 1 for block blk_-1608999687919862906 terminating
✗ Receiving block blk 7503483334202473044 src: /10.251.215.16:55695 dest: /10.251.215.16:50010
```

Fig. 9. Grouping with Respect to Block(blk) ID only



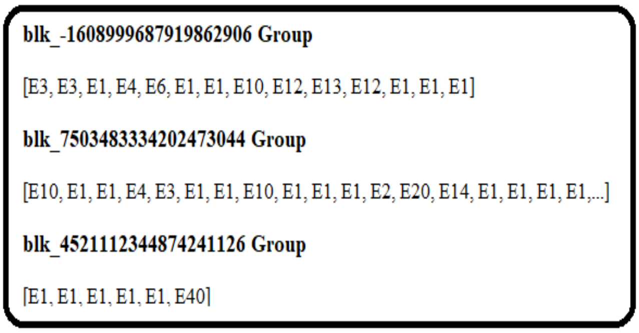
```
Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010
```

```
Receiving block <*> src: <*> dest: <*>
```

```
['blk_-1608999687919862906', '/10.250.19.102:54106', '/10.250.19.102:50010']
```

Fig. 10. Unstructured Log Message Comprises Log Event Template and Log Variables

The above image shows Unstructured raw data from the HDFS log is analysed with Drain to generate structured data as a log event template and log variables.



blk_-1608999687919862906 Group

[E3, E3, E1, E4, E6, E1, E1, E10, E12, E13, E12, E1, E1, E1]

blk_7503483334202473044 Group

[E10, E1, E1, E4, E3, E1, E1, E10, E1, E1, E1, E2, E20, E14, E1, E1, E1, E1,...]

blk_4521112344874241126 Group

[E1, E1, E1, E1, E1, E40]

Fig. 11. Log Messages are Grouped with Respect to blk id

The log data groups indicated by the HDFS sector identifiers are identified by log variables. The event sequence listings inside each block ID are formed in accordance with the image specifications, and log entries containing the similar block ID are clustered together.

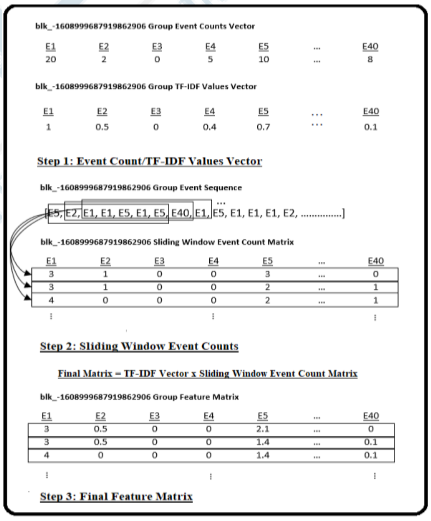
D. Feature Extraction of HDFS Data Set

Feature extraction is executed for each and every log message group according to the HDFS block ID. The steps for feature extraction are as follows:

Term frequency-inverse document frequency (event counts/TF-IDF): A bags-of-words method is used to compile event counts with each block ID group. To create a TF-IDF vector with each block ID, the maximum count of each occurrence for all blocks is also tallied.

Sliding window event counting: The collection of events included in each block ID is then divided by a sliding window that is then applied. The event counts inside each subset selection, each of which corresponds to a row in a matrix, are used to construct the block IDs.

Final Feature Matrix: After that, multiply the appropriate block ID TF-IDF vector by the block ID slide window event count matrix. In place of counting the number of events, this results in a detailed systematic on the TF-IDF values.



blk_-1608999687919862906 Group Event Counts Vector

E1	E2	E3	E4	E5	...	E40
20	2	0	5	10	...	8

blk_-1608999687919862906 Group TF-IDF Values Vector

E1	E2	E3	E4	E5	...	E40
1	0.5	0	0.4	0.7	...	0.1

Step 1: Event Count/TF-IDF Values Vector

blk_-1608999687919862906 Group Event Sequence

[E5, E2, E1, E1, E5, E1, E5, E40, E1, E5, E1, E1, E1, E2, ...]

blk_-1608999687919862906 Sliding Window Event Count Matrix

E1	E2	E3	E4	E5	...	E40
3	1	0	0	3	...	0
3	1	0	0	2	...	1
4	0	0	0	2	...	1
⋮			⋮			⋮

Step 2: Sliding Window Event Counts

Final Matrix = TF-IDF Vector x Sliding Window Event Count Matrix

blk_-1608999687919862906 Group Feature Matrix

E1	E2	E3	E4	E5	...	E40
3	0.5	0	0	2.1	...	0
3	0.5	0	0	1.4	...	0.1
4	0	0	0	1.4	...	0.1
⋮			⋮			⋮

Step 3: Final Feature Matrix

Fig. 12. Above Image Showing Feature Extraction in Three Steps.

E. Employee Activity Modelling Based on Daily Activity Summaries.

Data tables in this collection contain user behaviours for logon/logoff, Device, HTTP, file, and email. It is crucial to combine the behavioural data into a single flawless data table in chronologically in order to keep using the heterogeneous user behaviour data. Because the daily and weekly operations of the suggested inside threat detection models created in this study. Here, we combined a user's daily activity data that were previously fragmented and summarized them in order to subsequently weigh the strength of activity, which serves as an input variable to the detection model.

F. Employee Activity Modelling Based on Mouse operation

While the user wishes to choose or change the information stored over the cloud system, mouse operations like double clicking, dragging, and clicking take place here. Thus, the mouse movement's direction is classified. into the following categories: mouse drag (Mdrag), mouse double click (M-Dclick), mouse right clicking (Mrclick), mouse left click (Mlclick), and mouse double click (M-Dclick) [7].

Table 1. Operation of mouse

Movements of Mouse	***Distance***	***Speed***
M-Direction0	Present	Present
M-Direction1	Present	Present
M-Direction2	Present	Present
M-Direction3	Present	Present
M-Direction4	Present	Present
Mrclick	Present	Present
Mlclick	No	No
Mdrag	No	No

The N-grams approach is used to find out the distance. In the case of a distinct value, the query of the kth conduct is expressed as H = {H1, H2, …Hn}.and its frequency is expressed in Eq 1.

$$P_n(\mathrm{H}) = \{\nu \square \gamma\rho\alpha\mu|\ \nu \square \gamma\rho\alpha\mu = \left\langle \begin{matrix} h_i, \ldots \\ h_{i+1} \end{matrix} \right\rangle \iota\in\ [\mathrm{I}, \mathrm{N}+1-\nu] \qquad (1)$$

In order to determine the distance in the middle of two sets of data, the Jaccard coefficient is used.as follows:

$$\boldsymbol{Q_n(H_1, H_2) = 1 - \frac{|P_n(H_1) \cap_n^P (H_2)|}{|P_n(H_1) \cup_n^P (H_2)|}} \qquad (2)$$

Fig. 13. Distances between mouse

Distances between mouse locations are stated as movement characteristics, while mouse positions are recorded using two-dimensional coordinates (x, y). Movement characteristics like as angle, average distance, range and speed might reflect user personality characteristics.

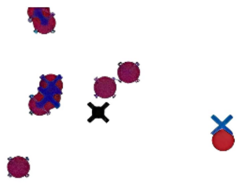


Fig. 14. Click movement

The click movement is composed by two movements: Press and Release. This means that a "click" action is created when a Press and a Release action are done twice in succession on the same point (x, y). Furthermore, the mapping method should be able to difference in the middle of the left and right mouse click buttons that is, the left mouse button's "click" as well as "double-click" and the right mouse button's "click" and "double-click". This allows for features such as the number of clicks, frequencies, and other relevant features to be recorded.

Fig. 15. Working of Press and drag button

When the "Pressed" button was engaged, the mouse did not instantly let go but rather slid a certain distance before being "Released," an action known as a "Drag Operation." This type of action is commonly used with mice and often goes unrecognized as part of the user experience.

Fig. 16. Scroll function with moving up or down.

Researchers tend to overlook the mouse "Scroll" function, but it can reveal people's recurring behaviour when operating a mouse. This action can be separated into two distinct parts - moving up or down.

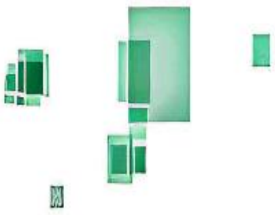


Fig. 17. Stay operation

Generally, researchers are finding mouse actions and clicking a mouse activity. However, they overlook the span between two mouse activities. This interval is also an "operation" of the mouse, referred to as the "Stay" operation. The "operation," which is shown as a semi-transparent square on the two-dimensional picture, can be referred to while there is none other mouse activity in one coordinate or the interval between two mouse actions. The size of the squares is employed to symbolize the period of stay.

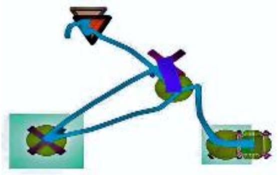


Fig. 18. Mouse behaviour

A picture showing the user's mouse behaviour (when m=100).

G. Employee Activity Modelling Based Keystroke operations.

While we hit the keys for a certain function, the keystroke events take place. They resemble KFunc (F1, F2...), KCtrl (ctrl keys, alt, win), KDir (key like ->, -), KNum (comprises number keys 1, 2, 3...), KShift (Shift+num, alpha), KAlpha (Alphabets, such as A, a, b, c...), and KOhter (tab, caps lock...). Duration is one of the fundamental event features [7]. As a result, the time signifies is as the interval between the keystroke and key release times. namely, as follows:

$$\Delta\upsilon\rho\alpha\tau\iota o\nu\ (\upsilon\sigma\varepsilon\rho, \mathrm{H}) = time_f(\varpi, \mathrm{H}) - time_s(\varpi, \mathrm{H}) \qquad (3)$$

The time taken by a user v to press a key H is denoted as time p(v, H), while the time taken to release the same key H is represented as timer(v, H).The delay of the keyboard that is not being utilized is defined in Eq 4.

$$\mathrm{N}\ (\varpi, \mathrm{H}) = \boldsymbol{time_s}(\varpi, \mathrm{H}) - \boldsymbol{time_p}(\varpi, \mathrm{H}). \qquad (4)$$

The characteristics of these keystroke activities are described in Eq (5).

$$\begin{Bmatrix} \langle \boldsymbol{duratuion}(\boldsymbol{v}, \boldsymbol{H_1}), \boldsymbol{latency}(\boldsymbol{v}, \boldsymbol{H_1})\rangle, \dots \dots \\ \langle \boldsymbol{duration}(\boldsymbol{v}, \boldsymbol{H_n}), \boldsymbol{latency}(\boldsymbol{v}, \boldsymbol{H_n})\rangle \end{Bmatrix} \qquad (5)$$

To find possible dangerous insiders in cloud, a deep learning-based classifying algorithm is used to gather data about user activities such clicks, keystrokes, and mouse movements [7].

H. Employee Modelling Activity Using Email Operations.

A table in the cert dataset stores the daily email use records of an employee, including the number of emails sent and received. Here, the email data table from the HDFS as well as CERT datasets additionally includes the email's content records in addition to its log records. Because sender/receiver data is also obtained from email log records with clustered weighted features, a third form of user activity evaluation for insider threat identification, we built the email network communication on a weekly basis. Here, "abc" is a fictitious firm name for the CERT and HDFS datasets, and @abc.com is a valid email address. Employees in this dataset either utilized their company-issued email addresses (@abc.com) or other addresses (@gmail.com, @yahoo.com, @titan.com, @hubspot.com, @zohomail.com, @icloud.com, @outloot.com). Users sent as well as received emails to or from persons inside the same department, from outside firm, or in various departments within the same organization. Therefore, an employee's personal email is used as a component in this research, and the interactions among two email accounts are measured based on the volume of arriving and departing emails.

I. A Structured Approach for Training RBMs

Here, we provide a methodical procedure for selecting the training parameters for a given equipped RBM. Due to the difficulty of guaranteeing the process of learning convergence and for the purpose of preventing overfitting, a modelled method is essential in the setting of RBMs. By utilizing a CERT and HDFS dataset, we want to demonstrate the viability of our technique in the context of network malware detection at the final step. Therefore, the method consists of three main components: weight initiation, pre-training, and also fine-tuning, along with a few extra options for parameterization.

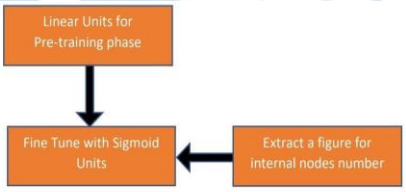

Linear Units for Pre-training phase
Fine Tune with Sigmoid Units
Extract a figure for internal nodes number

Fig. 19. Structured Approach for Training RBMs

J. The Three Major Aspects

The amount for fine-tuning may be calculated using layer-wise unsupervised pre-training. Here, RBM is training for a predetermined number of epochs for each pair of neighbouring layers. A network is trained with a set of samples during an epoch. After pre-training, fine-tuning is performed. In this case, one of the fine-tuning functions is used, such as describe the number of Hidden Nodes to use in measuring the RBM internals and selecting the best choice that use the accuracy maximization criterion [17].

IV. FLOW CHART

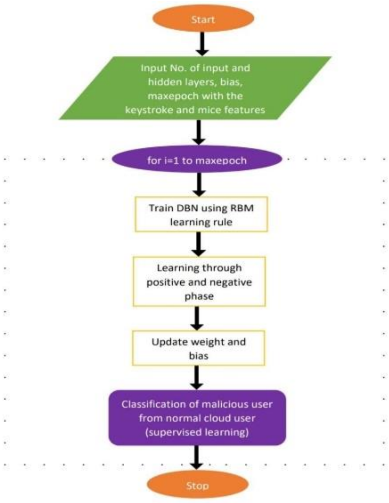

Start
Input No. of input and hidden layers, bias, maxepoch with the keystroke and mice features
for i=1 to maxepoch
Train DBN using RBM learning rule
Learning through positive and negative phase
Update weight and bias
Classification of malicious user from normal cloud user (supervised learning)
Stop

Fig. 20. Flow Chart

V. RESULTS

A CNN model is used by the log anomaly detection project to find aberrant log data. The steps used by the log anomaly detection are as follows:

- **Parse**: The process of converting unstructured log data into a format that includes a log event template and log variables.
- **Feature Extraction:** To create feature matrices, use TF-IDF on event counts and sliding windows.

Note on TF- IDF:

Term Frequency-Inverse Document Frequency is also known as TF-IDF. Representing the importance of a word or phrase within a given text is one of the most important strategies for information retrieval.

- **Log Anomaly Detection Model:** A CNN model trained on log data with Restricted Boltzmann Machine algorithm implemented in Deep Belief Network and fed inputs from feature matrices.

The log anomaly detection model was assessed using HDFS and CERT log data, and test set precision, recall, as well as F-score scores all were higher than 99%.

Consists of the following architectural components:

- Two MLP (Multilayer Perceptron) hidden layers (120 and 84 nodes)
- Two convolutional layers (16 and 32 filters with 2x2 kernels each) with max pooling (2x2 kernels)
- Two MLP output layers (2 nodes each representing normal and anomalous labels)
- The convolutional and MLP layers use ReLU (Rectified Linear Unit) activation with Restricted Boltzmann Machine algorithm and the output layer uses Softmax.

A. What is the ReLU activation and Softmax Function?

ReLU activation:

Rectified linear unit is what it stands for. The most common activation function is this one. primarily used in neural network's hidden layers. Equation: A(x) = max (0, x). If x is positive, it outputs x; if not, it outputs 0. Due to the non-linear of nature, we may simply backpropagate errors to activation of several layers of neurons using Relu function.

Softmax Function:

The softmax function may be used to change the real values of matrix S to a matrix of S true values that add to 1. The softmax converts input values that which could be positive, minus, zero, or greater than one into values ranging from 0 and 1, which may be interpreted as probabilities. While it will constantly fall between 0 and 1, the softmax translates small or negative values into small probability and large or positive values into high probabilities.

The softmax formula is as follows:

$$\sigma(\vec{Z})i = \frac{e^{Z_i}}{\sum_{j=1}^{S} e^{Z_j}} \qquad (11)$$

Mathematical definition of the softmax functionality distribution.

where each zi value, which is a part of the input vector, can be any real value. The probability density function is legitimate because the normalization factor at the bottom of a calculation ensures that perhaps the function's extracted features will all add up to 1.

B. Training Model

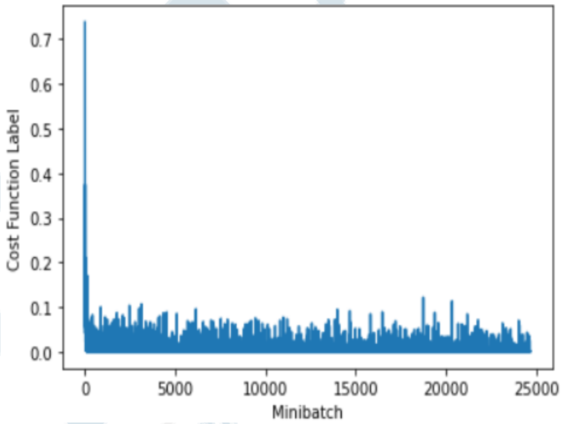


Fig. 21. Minibatch vs Cost Functional Label

The data in the Fig. 21. graph is plotted between a 0.7 Cost Fully functioning label as well as a Mini-batch size of up to 25000. (CFL). CFL is quite great in the initially minibatch before decreasing to 0.1 CFL over time.

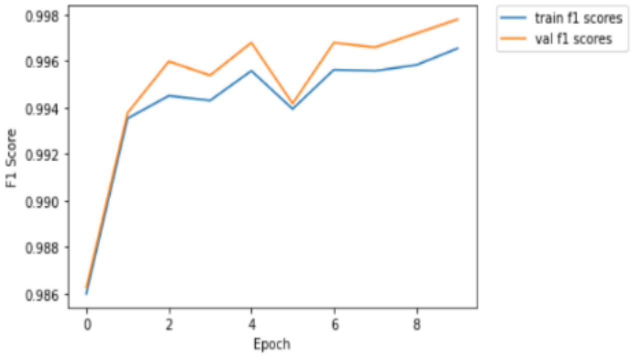


Fig 22. Epoch vs f1 score

The values in the Fig. 22. graph is shown between 9 epochs but also data with an F1 score up to 0.998. The reliability score obtained from the R-square test was 0.86625. The blue line represents the Train F1 score, whereas the orange line represents the Val F1 score. With only a 0.002 difference, the two f1 scores are nearly identical.

C. Training and Evaluation

The shallow CNN architecture of the log anomaly model contains two convolutional layers and two max pooling layers. Two multi-perceptron hidden layers receive the output from the last max pooling layer. Two nodes that represent the anomalous and normal labels make up the final layer. The block ids in the HDFS log dataset, which are classified either as normal or anomalous and CERT Unlabelled as either normal or anomalous was used to train the model. Both HDFS as well as CERT data set was used to assess the model. The train/test split for the both the datasets was 80/20. The test data were used to evaluate the model after it had been trained using the labelled HDFS data and unlabelled CERT dataset.

The following tables provide the model results. The measurements show that utilizing both the log dataset, the log anomaly detection procedure is functioning quite well.

D. Categorization of Training

Table 2. Categorization of training

	Absolute Normal	*Absolute Anomalous*
Regular Mode	305731	22
Model Absurdity	47	9808

E. Categorization of Testing

Table 3. Categorization of TESTING

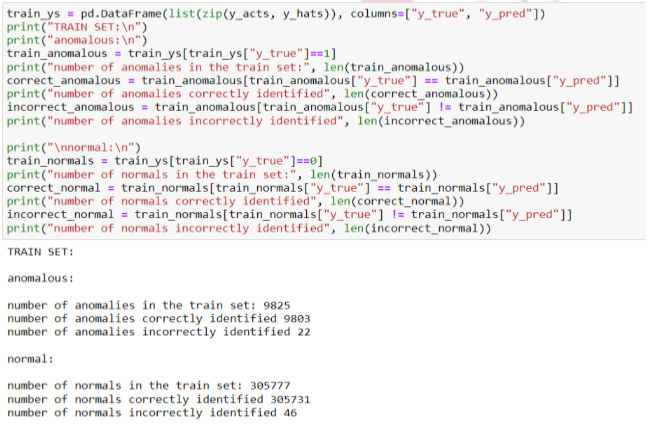
	Absolute Normal	*Absolute Anomalous*
Regular Mode	118553	5
Model Absurdity	2	1977

F. Measures of Model Performance

Table 4. MEASURES OF MODEL EXECUTION

	Precision	*Recall*	*F-score*
Training	*99.3*	*99.7*	*99.6*
Testing	*99.8*	*99.5*	*99.4*

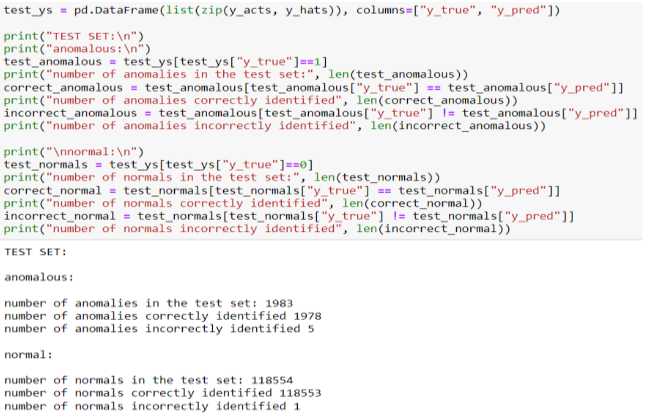
G. Results Snapshots



```
train_ys = pd.DataFrame(list(zip(y_acts, y_hats)), columns=["y_true", "y_pred"])
print("TRAIN SET:\n")
print("anomalous:\n")
train_anomalous = train_ys[train_ys["y_true"]==1]
print("number of anomalies in the train set:", len(train_anomalous))
correct_anomalous = train_anomalous[train_anomalous["y_true"] == train_anomalous["y_pred"]]
print("number of anomalies correctly identified", len(correct_anomalous))
incorrect_anomalous = train_anomalous[train_anomalous["y_true"] != train_anomalous["y_pred"]]
print("number of anomalies incorrectly identified", len(incorrect_anomalous))

print("\nnormal:\n")
train_normals = train_ys[train_ys["y_true"]==0]
print("number of normals in the train set:", len(train_normals))
correct_normal = train_normals[train_normals["y_true"] == train_normals["y_pred"]]
print("number of normals correctly identified", len(correct_normal))
incorrect_normal = train_normals[train_normals["y_true"] != train_normals["y_pred"]]
print("number of normals incorrectly identified", len(incorrect_normal))
```

```
TRAIN SET:

anomalous:

number of anomalies in the train set: 9825
number of anomalies correctly identified 9803
number of anomalies incorrectly identified 22

normal:

number of normals in the train set: 305777
number of normals correctly identified 305731
number of normals incorrectly identified 46
```

Fig. 23. Train Set Results



```
test_ys = pd.DataFrame(list(zip(y_acts, y_hats)), columns=["y_true", "y_pred"])

print("TEST SET:\n")
print("anomalous:\n")
test_anomalous = test_ys[test_ys["y_true"]==1]
print("number of anomalies in the test set:", len(test_anomalous))
correct_anomalous = test_anomalous[test_anomalous["y_true"] == test_anomalous["y_pred"]]
print("number of anomalies correctly identified", len(correct_anomalous))
incorrect_anomalous = test_anomalous[test_anomalous["y_true"] != test_anomalous["y_pred"]]
print("number of anomalies incorrectly identified", len(incorrect_anomalous))

print("\nnormal:\n")
test_normals = test_ys[test_ys["y_true"]==0]
print("number of normals in the test set:", len(test_normals))
correct_normal = test_normals[test_normals["y_true"] == test_normals["y_pred"]]
print("number of normals correctly identified", len(correct_normal))
incorrect_normal = test_normals[test_normals["y_true"] != test_normals["y_pred"]]
print("number of normals incorrectly identified", len(incorrect_normal))
```

```
TEST SET:

anomalous:

number of anomalies in the test set: 1983
number of anomalies correctly identified 1978
number of anomalies incorrectly identified 5

normal:

number of normals in the test set: 118554
number of normals correctly identified 118553
number of normals incorrectly identified 1
```

Fig. 24. Test Set Results

Both the Fig. 23 and Fig.24 shows train and test set results showing the number of anomalies and normal cloud users are there, among those which of them are correctly identified and which of them are incorrectly identified.

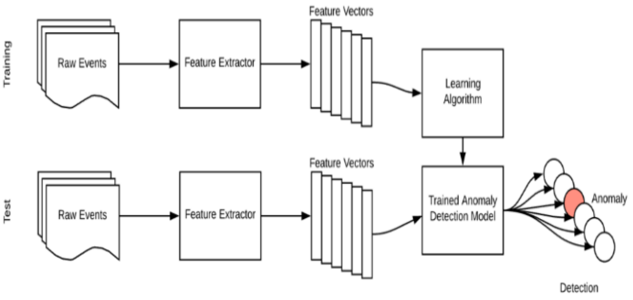


Fig. 25. Workflow of Anomaly Detection Before Alerting

VI. CONCLUSION

This study, likely to do well at detecting insider attacks. Internal attackers frequently exhibit exceptional intelligence by disguising themselves as trustworthy authorities. we assessed an RBM's suitability as a machine learning model. The purpose of this paper was to detect malicious activity using machine learning techniques. We proposed a system pipeline that is capable of identifying malicious users accurately and can adapt to various organizational composition as well as modifications to the structure of the log file. We improved insider threat detection with the use of an automated deep belief neural network.

Our endeavor is able to warn the system of a potential internal threat. It is impossible to totally prevent it, but we can contain it to some level with technical regulations like safety measures, firewalls, and other safeguard apps. The model is made to discover internal dangers, from gathering and preparing data to examining that data with ML methods. It will classify the anomaly and alert the system regarding the insider threat and recognize malevolent activity that is done inside the company. In conclusion, the goal of our project is to make a user-friendly software program that can identify the internal threat and warn the system, as well as spot malicious conduct taking place within the organization.

REFERENCES

[1] Mehreen Afzal, Rabia Latif, Waseem Iqbal, “Behavioral Based Insider Threat Detection Using Deep Learning”, IEEE, 2021

[2] P. Varsha Suresh, Minu Lalitha Madhavu, “Insider Attack: Internal Cyber Attack Detection Using Machine Learning”, IEEE, 2021

[3] Sashi Kumar Mamidanna, C R K Reddy, Akash Gujju, “Detecting an Insider Threat and Analysis of XGBoost using Hyperparameter tuning”, IEEE, 2022

[4] Madhu Raut Sunita Dhavale, Amarjit Singh, Atul Mehra, “Insider Threat Detection using Deep Learning: A Review”, IEEE, 2020

[5] Tamer Aldwairi, Dilina Perera, MarkA.Novotny "An evaluation the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection" july 2018

[6] T. O. Oladimeji1, C. K Ayo1 and S.E Adewumi1, “Review on Insider Threat Detection Techniques””, 2019

[7] S. Anakath, R. Kannadasan, Niju P. Joseph, P. Boominathanand, G. R. Sreekanth, “Insider Attack Detection Using Deep Belief Neural Network in Cloud Computing”, 2021

[8] Aaron Tuor, Samuel Kaplan, Nicole Nichols and Sean Robinson and Brian Hutchinson, “Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams”. 2017

[9] Dr.Mohammed Dosh, “Detecting insider threat within institutions using CERT dataset and different ML technique”, 2021

[10] Teng Hu, Weina Niu, Xiaosong Zhang, Xiaolei Liu, Jiazhong Lu,1and Yuan Liu2, “An Insider Threat Detection Approach Based on Mouse Dynamics and Deep Learning”,2021

[11] Folasade Mercy Okikiola, Abiodun Muyideen Mustapha, Adeniyi Foluso Akinsola, Michael Adio Sokunbi “A New Framework for Detecting Insider Attacks in Cloud-Based E-Health Care System”, IEEE, 2020

[12] T. Nathezhtha, V. Yaidehi, “Cloud Insider Attack Detection Using Machine Learning”, IEEE, 2018

[13] Morshed U. Chowdhury, Biplob Ray, Sujan Chowdhury Sutharshan Rajasegarar, "A Novel Insider Attack and Machine Learning Based Detection for the Internet of Things" July 2021

[14] Shuhan Yuan, Xintao Wu, "Deep learning for insider threat detection: Review, challenges and opportunities" May 2021

[15] Mittal, Anupam Garg, Urvashi "A review for insider threats detection using machine learning" October 2022

[16] Konstantinos Demertzis, Lazaros Iliadis, Elias Pimenidis, Panagiotis Kikiras, "Variational restricted Boltzmann machines to automated anomaly detection" 01 March 2022

[17] Arnaldo Gouveia, Miguel Correia "A Systematic Approach for the Application of Restricted Boltzmann Machines in Network Intrusion Detection" 18 May 2017

[18] Duc C. Le and A. Nur Zincir-Heywood “Machine learning based Insider Threat Modelling and Detection” 2019

[19] D. C. Le, N. Zincir-Heywood, "Anomaly Detection for Insider Threats Using Unsupervised Ensembles," in IEEE and Service Management, vol. 18, no. 2, pp. 1151–1163. June 2021.

[20] Efthimios Pantelidis, Gueltoum Bendiab, Stavros Shiaeles, Nicholas Kolokotronis "Insider Detection using DeepAutoencoder and Variational Autoencoder Neural Networks" July 2021

[21] D. C. Le, N. Zincir-Heywood and M. I. Heywood, "Analyzing Data Granularity Levels for Insider Threat Detection Using Machine Learning," in IEEE and Service Management, vol. 17, no. 1, pp. 29-45, March 2020,

[22] Kim, M. Park, S. Cho, H. Kim and P. Kang, “Insider threat detection based on user behavior modeling and anomaly detection algorithms,” Applied Sciences, vol. 9, no. 19, pp. 1–21, 2019.

[23] Homoliak, J. D. Guarnizo, Y. Elovici, F. Tofalini and M. Ochoa, “Insight into insiders: A survey of insider threat taxonomies, analysis, modeling, and countermeasures,” ACM Computing Surveys, vol. 52, no. 2, pp. 1–40, 2019.

[24] Misbah Anwer, Ghufran Ahmed, Adnan Akhunzada, Shahbaz Siddiqui, "Intrusion Detection Using Deep Learning". IEEE, 2021.

[25] Hongyu Liu, Bo Lang "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey" October 2019.