BEHAVIOURAL BASED INSIDER THREAT DETECTION USING DEEP LEARNING

FINAL YEAR PROJECT REPORT

Submitted by

ABDUL JAVAZ T	**: TJE19CS002**
ADARSH K P	**: TJE19CS006**
ADVAITH S KRISHNA	**: TJE19CS008**
DENNIS DANIEL	**: TJE19CS030**

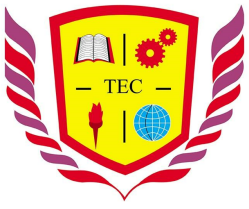
in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING



THEJUS ENGINEERING COLLEGE, VELLARAKKAD

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

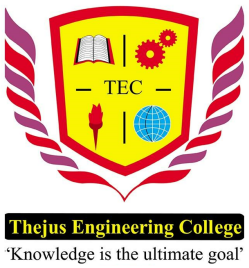
June 2023

(http://www.thejusengg.ac.in)

Department of Computer Science and Engineering
Thejus Engineering College

Vellarakkad, Thrissur - 680 584

(http://www.thejusengg.ac.in)



Certificate

Certified that this Project titled **"BEHAVIOURAL BASED INSIDER THREAT DETECTION USING DEEP LEARNING"** is the bonafide work of **ABDUL JAVAZ T, ADARSH K P, ADVAITH S KRISHNA, DENNIS DANIEL** of final year B.Tech in partial fulfillment of the requirement for the award of Bachelor of Technology Degree in Computer Science and Engineering awarded by the APJ Abdul Kalam Technological University, during the academic year 2022-2023under my guidance.

Project Guide

Ms.Sneha Johnson

Asst. Prof., Dept. of CSE

Head of Department

Dr.Suresan Pareth

Prof., Dept. of CSE

ACKNOWLEDGEMENT

The project on topic "BEHAVIOURAL BASED INSIDER THREAT DETECTION USING DEEP LEARNING" is taken as a part of the curriculum for the award of B.Tech degree in Computer Science and Engineering.

During the course of the project work, several persons collaborated directly and indirectly with us. Without their support it would be impossible for us to finish our work. That is why we wish to dedicate this section to recognize their support.

We sincerely express our gracefulness to Dr.Vijayan P, Principal of Thejus Engineering College, Vellarakkad, for providing all the necessary facilities.

We take this opportunity to extend our sincere regards to Dr.Suresan Pareth, Head of Computer Science/ our project guide, for encouraging and supporting us throughout the project and also for taking keen interest in out project work. Through his expert supervision, encouragement and construction criticism gave us constant support amidst his busy schedule.

I am grateful to express my thanks to all the faculty members of our department for their support. I articulate my gratitude to all my friends for their support and help for this work.

Last, but not the least I wish to express my gratitude to God Almighty for His abundant blessings without which this effort would not have been successful.

June 2023

ABSTRACT

The most detrimental cyber attacks are usually not originated by malicious outsiders or malware but from trusted insiders. The main advantage insider attackers have over external elements is their ability to bypass security checks and remain undiscovered, this may cause serious damage to the organizational assets. This paper focuses on insider threat detection through behavioral analysis of users. User behavior is categorized as normal or malicious based on user activity. A series of events and activities are analyzed for feature selection to efficiently detect adversarial behavior. Selected feature vectors are used for model training during the implementation phase. A deep learning based approach is proposed that detects insiders with greater accuracy and low false positive rate. A rich event / user role based feature set containing Logon/Logoff events, Userrole, Functional unit etc are used for detection.

Contents

List of Figures

List of Tables

LIST OF SYMBOLS AND ABBREVIATIONS

DL	Deep Learning
URL	Uniform Resource Locator
RAM	Random Access Memory
UI	User Interface
VSC	Visual Studio Code

Chapter 1

INTRODUCTION

1.1 OVERVIEW

One of the most basic, yet hard to solve problem in cyber security is the identification of adversarial behavior. The exploitation and leakage of sensitive data and information by malicious insiders is getting worse day by day. According to "Insider Report 2018" 90 percentage of the organizations are prone to insider attack. Around 60 percentage organizations encountered one or more insider attacks in 2019. Since an insider has authorized access to an organization assets, therefore they might have better opportunity to undermine the confidentiality, availability or integrity of data than an external attacker. Various primary and secondary elements that may serve as an inspiration for an insider includes financial gain or greed, revenge, anger, thrill, pressure, treachery, discontentment, jealousy, organizational politics and acknowledgement.

1.2 OBJECTIVE

The objective of insider threat detection through behavioral analysis of users. User behavior is categorized as normal or malicious based on user activity. A series of events and activities are analyzed for feature selection to efficiently detect adversarial behavior. Selected feature vectors are used for model training during the implementation phase. A deep learning based approach is proposed that detects insiders with greater accuracy and low false positive rate. A rich event / user role based feature set containing Logon/Logoff events, User role, Functional unit etc are used for detection.

1.3 PROBLEM STATEMENT

An efficient intrusion detection system must accurately identify all threats even if they form a small fraction of the intrusion data. The effect of class imbalance on the KDD-CUP-99 dataset is evaluated using four popular classification techniques and the results are analyzed. The problem is to identify whether a network connection is good or malicious. We are building a predictive model distinguishing intrusions or attacks from normal connections. The data set contains nine weeks of TCP dump data prepared with multiple attacks. Each connection is labelled as 'normal' or as the attack type.

Chapter 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEMS

There are several existing Insider Threat Detection techniques that are widely used . Here are some notable techniques:

2.1.1 USER BEHAVIOR BASED INSIDER THREAT DETECTION TECHNIQUES

In user behavior based detection, user behavior is categorized in to two types, that is normal and malicious. Each user behavior is logged and is compared with a standard rule set (created by experts). If the behavior deviates from normal, it is considered malicious. A supervised time series based solution using two layer deep auto-encoder can also be used for insider attack detection. A technique using LSTM-CNN algorithm has been shown to identify user anomalous behavior in, by monitoring user activities and extracting temporal features. While in [6], detection algorithm XGBoost has been used and behavior characteristic features are extracted from audit logs. Technique proposed in extracted features and fields from user behavior logs for behavior auditing, and then these log files are used to train the Improved Hidden Markov Model (IHMM) for detection of malicious behavior. Random forest algorithm can be used for behavior analysis of individual user by analyzing its activities over a period of time. A user sentiment profile has been designed in to give a prediction scheme using user's network browsing content and emails. A framework known as "Insider Catcher" is prposed in. The proposed framework uses LSTM, a deep learning technique to model system logs as an organized sequence. Work

done in uses distance measurement techniques (DL distance, Jaccard Distance and Cosine Distance) through analysis of user activity for insider detection. Kernel PCA and LSTM-RNN for insider detection is proposed.

2.1.2 GRAPH BASED INSIDER THREAT DETECTION TECHNIQUES

With the technological advancement, user's data is now heterogeneous and multi dimensional. The heterogeneous data generated from various sources consists of network activity, psychological factors, organizational dynamics, employee behavior etc. the data thus forming structural patterns. For detecting insider threat in this complex, structural and heterogeneous data, graph based approach is used. Technique proposed in detects the malicious conduct of an employee based not only on its own activities but also on the malicious activities of the employees with same job roles. Prospective insiders are recognized by designing a graph signal processing technique. Employees normal data usage patterns are reported and compared to identify anomaly. An insider attack detection mechanism using Gaussian Mixture model is proposed which included security expert knowledge and other non-technical indicators of insider threat as key elements of the system. A graph analysis and anomaly detection based hybrid framework, which consists of two modules "Graphical Processing Unit" (GPU) and "Anomaly Detection Unit" (ADU) for insider threat detection has been given. Attributed graphs for showing high dimensional, diverse data are used for insider threat detection. A framework which uses the combined approach of Structural Anomaly Detection (SA) and Psychological Profiling (PP) of users for insider threat detection is proposed.

2.1.3 INSIDER DETECTION USING OTHER TECHNIQUES

A network based insider attack flexible approach "Gargoyle" is proposed. The trustworthiness of the context of an access request is evaluated through a new set of contextual attributes called Network Context Attribute (NCA), information such as the user's device capacity, security-level, network connection status etc. are obtained from network traffic analysis. Network packet inspection has been used for insider threat detection, while honey pot sensors have been used within the company's local network to detect

insiders. The model proposed in [23] works by properly designing rules and regulations into complex events and investigating whether employees conduct conforms to the rules and regulations. To predict and detect insider threat, disturbing psychological patterns of individual users are obtained by analyzing electronic communications in [32]. A state machine system is proposed that can efficiently integrate policies from rule based anomaly detection systems in order to create models which are followed by the insiders to launch an attack.

2.1.4 A COMPARATIVE STUDY OF EXISTING TECHNIQUES

The overview and comparison shown in the Table 1 2 respectively gives a clear picture of various ML and DL techniques used for insider threat detection. Some of the techniques are very efficient, but have some deficiencies in terms of complexity, performance evaluation metrics and evaluation dataset. Some models are not evaluated on real life scenarios and are processing memory intensive. Some have relatively small test data, which does not fully evaluate the performance of the technique.

It is also observed that most Role based or Behavior based techniques produce significant quantitative results as compared to graph based and other techniques. The most widely used technique is LSTM. Another widely used technique is Deep AutoEncoders, it has the ability to be used on real valued datasets and are quick concise. Keeping in view the above discussion, a novel hybrid DL approach has been designed in this research to detect insiders efficiently, with low processing and memory requirements, with low false positive rate and higher accuracy.

2.2 LITERATURE REVIEW

Insider threats pose a critical challenge for securing computer networks and systems. They are malicious activities by authorised users that can cause extensive damage.The solution for it is the Anomaly detection and classification models, to detect insider threats.The drawback is Compared to the anomaly detection methods, the ML classification models produced the best overall results.

According to IBM 2016 Cyber Security Intelligence Index among all the security breaches or attacks recorded in 2015 worldwide, more than 60 percentage are caused by

insiders only.Proposed a scenario-based insider threat detection approach from human behavioral activities. Comparitively less accuracy

User behavior is categorized in to two types, that is normal and malicious Each user behavior is logged and is compared with a standard rule set (created by experts). If the behavior deviates from normal, it is considered malicious.Comparitively less accuracy

Graph based insider threat detection techniques.With the technological advancement, user's data is now heterogeneous and multi dimensional. With the technological advancement, user's data is now heterogeneous and multi dimensional. But it is Time consuming

Insider threat detection using other techniques.A network based insider attack flexible approach.The trustworthiness of the context of an access request is evaluated through a new set of contextual attributes called Network Context Attribute (NCA), information such as the user's device capacity, security-level, network connection status etc. are obtained from network traffic analysis.Less accuracy.

Detection and prediction of insider threats to cyber security.Data integrity is another important security concern. Damage to data integrity can often cause more serious problems than confidentiality breaches. Proposed a insider threat detection approach from human behavioral activities. But it is time consuming,Less accuracy.

2.3 PROPOSED SYSTEM

The proposed system using "LSTM-Autoencoder" for insider threat detection. Our proposed approach consists of multiple stages. At first user data is gathered from various csv files, the data is then processed and a rich event /user role based feature set containing Logon/Logoff events, user role, user functional unit, user department, user activity etc are extracted. Selected features are then used for model training and evaluation.

Insider threat dataset has a multivariate time-series data which consists of several variables discovered over a time interval. On this multivariate time-series data for insider attack detection an LSTM Autoencoder has been built.

LSTM is very effective in natural language processing. It can automatically learn features. It is very good in processing sequence and time series data due to which it is

used to model user behavior. While autoencoder has the ability to be used on real valued datasets and are quick concise.

LSTM auto-encoders are explicitly designed to avoid the long-term dependency problem. Remembering information for long period of time is practically their default behavior and hence they have an advantage over normal auto-encoders. So it is one of the best technique for finding anomalies in time series data.

LSTM is used for model training. As LSTM are designed to look at the historical data to make predictions, it processes data up to (t-lookback) to make a prediction at a given time t. It takes a 3D array as input. samples × lookback × features Once the data is ready, it is divided into the train data and test-data. Train-data is used for model training and parameter tuning, while test-data is used for model evaluation.

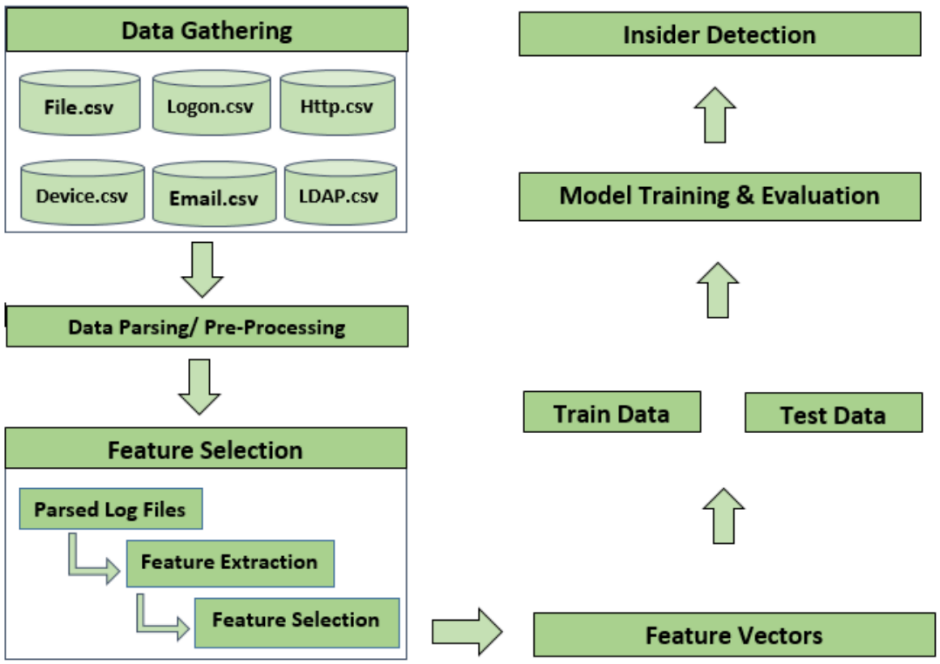


Figure 2.1: Structure of our proposed solution

Table 2.1: LITERATURE SURVEY

PAPER NAME	PROBLEMS	SOLUTION	DRAWBACK
Insider Threat Detection Using Machine Learning Approach by Bushra Bin Sarhan and Najwa Altwaijry	Insider threats pose a critical challenge for securing computer networks and systems.They are malicious activities by authorised users that can cause extensive damage	Anomaly detection and classification models, to detect insider threats	Compared to the anomaly detection methods, the ML classification models produced the best overall results.
Scenario based insider threat detection from cyber activities by Pratik Chattopadhyay,Lipo Wang and Yap-Peng Tan	According to IBM 2016 Cyber Security Intelligence Index among all the security breaches or attacks recorded in 2015 worldwide, more than 60	Proposed a scenario-based insider threat detection approach from human behavioral activities.	Comparitively less accuracy
Insider threat detection using other techniques	A network based insider attack flexible approach	The trustworthiness of the context of an access request is evaluated through a new set of contextual attributes called Network Context Attribute (NCA), information such as the user's device capacity, security-level, network connection status etc. are obtained from network traffic analysis	Less accuracy

PAPER NAME	PROBLEMS	SOLUTION	DRAWBACK
User behaviour based insider threat detection techniques	User behavior is categorized in to two types, that is normal and malicious	Each user behavior is logged and is compared with a standard rule set (created by experts). If the behavior deviates from normal, it is considered malicious	Comparitively less accuracy
Graph based insider threat detection techniques	With the technological advancement, user's data is now heterogeneous and multi dimensional.	With the technological advancement, user's data is now heterogeneous and multi dimensional.	Time consuming
Detection and prediction of insider threats to cyber security	Data integrity is another important security concern. Damage to data integrity can often cause more serious problems than confidentiality breaches.	Proposed a insider threat detection approach from human behavioral activities.	Less accuracy

Chapter 3

SYSTEM DESCRIPTION AND OPERATION

3.1 SYSTEM ARCHITECTURE

3.1.1 DATA GATHERING

The dataset used is the CMU CERT synthetic insider threat dataset r4.2. The dataset consists of synthetic data of both normal and malicious insiders. To make the approach simple all the csv files are aggregated and most relevant features are extracted. The dataset consists of 1000 synthetic users out of which 70 are malicious insiders. The dataset consists of various csv files, in which following are included:

1) logon.csv: Log of users logging in and out on a computer

2) device.csv:Log of users connecting and disconnecting external devices (USB)

3) http.csv:Users browser history

4) email.csv:Email logs

5) file.csv:Log of user activity on files (coping file to an external device)

6) psychometric.csv:Contains user personality attributes i-e. OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism).

7) LDAP (Lightweight Directory Access Protocol):Set of files describing all users and their assigned job roles.

Total number of rows are 32,770,220. The reason for selecting version r4.2 is that most datasets had one instance of each scenario. Dataset 4.2 was a “dense needle” dataset and had many instances of each scenario.

3.1.2 INSIDER THREAT SCENARIO

In this dataset, malicious insider user is designed to accomplish one out of the following two scenarios at some point in time.

1) Use of external hard drives, or work after hours, login activity after office hours by the user who did not have such previous routine, using of a flash drive, uploading data to wikileaks.org and then leaving the organization shortly thereafter.

2) User visiting job sites and seeking employment from a competitor. Use of a flash drive (at markedly higher rates than their previous activity) to steal data before leaving the organization.

3.1.3 DATA PRE-PROCESSING

The dataset consists of various csv files (logon, file, HTTP, email, device, LDAP) and each file contained raw data. This data cannot be fed to the algorithm and it needs to be preprocessed. All the csv files are parsed and an aggregated csv file i-e. Master file was created which contained data from all csv files. A features set was extracted from this aggregated master file containing both integers and text strings. The values will have to be suitably encoded to be used as input for our proposed algorithm. There were some missing data in the synthetic dataset 4.2, due to which it does not look like real life data collected from sensors. The algorithms cannot work with this missing data. So at this step data is pre-processed by replacing missing values with the estimated mean value of the relevant feature. The machine learning algorithms uses numerical or integer values so the values are integer encoded. Data pre-processing is a tedious job, so a code was written to automate this proves in order to perform the job efficiently and save time and resources

3.1.4 FEATURE EXTRACTION

All the CSV files are parsed and relevant data fields are identified that can efficiently learn and predict user behavior. Moreover, it depends on the insider scenarios with which we are dealing. For example, a user normal working hours are from 8AM to 7PM, so it is normal if a user login and logout during this time. However, if a user login after office hours, use some flash drive or USB and leaves shortly thereafter, this behavior is considered malicious.Psychometric.csv is not used in feature selection. Features from all other csv files included integer encoded day, time, pc, user id, PC, user role,user functional unit, user department and activity features. Id of the features is redundant and is not included

Most previous approaches used fixed time window based features, however it may reduce the likelihood of detecting anomalous behavior. Instead of using fixed time based windows, user session based flexible time window is used. A user session includes various activities i-e. it starts with logon activity followed by other activities (http,email,file etc.) and ends with logoff activity. Day, time, user id, PC and activity feature reveals user session based activity and information. However, user role, functional unit and department reveals important information related to user job role and responsibilities, and are populated against each user.

The collected features from various csv files contains multiple categorical and ordinal values given as text strings, and will not be used as input for our algorithm. Therefore, the values will have to be suitably encoded for the algorithms to make correct prediction. Presence of a feature is represented by “1” while the absence by “0”.

Features	**Values**
Day	0-6
Time	1-24
Activity	1-7
User_id	1-1000
User_role	1-42
User_functional_unit	1-6
User_department	1-7
PC	Unique number

Figure 3.1: Feature values

Activity	Label
Logon	1
Logoff	2
Connect	3
Disconnect	4
E-mail	5
File	6
Http	7

Figure 3.2: Activity labels

User_Functinal_Unit	Label
Administration	1
Research And Engineering	2
Manufacturing	3
Finance	4
Sales And Marketing	5
Purchasing And Contracts	6

Figure 3.3: User functional unit encoding

3.1.5 DATA SET

The dataset used is the CMU CERT synthetic dataset r4.2 which consists of 1000 synthetic users out of which 70 are malicious insiders. The dataset contains only 0.03 percentage anomalous and 99.7 percentage normal instances. The dataset is divided into train, valid, and test data for model training, validation and testing. The proportion in which data is divided includes: 70 percentage training data, 10 percentage validation data and 20 percentage testing data. Total number of logs are 32,770,220 out of which 17,193 are trainable parameters.

dataset used is the CMU CERT synthetic dataset r4.2 which consists of 1000 synthetic users out of which 70 are malicious insiders. The dataset contains only 0.03 percentage anomalous and 99.7 percentage normal instances. The dataset is divided into train, valid, and test data for model training, validation and testing. The proportion in which data is divided includes: 70 percentage training data, 10 percentage validation data and 20 percentage testing data. Total number of logs are 32,770,220 out of which 17,193 are trainable parameters.skip

3.2 PHASES

1. Data set Collection: The dataset collection is very important for the project.It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the ”signature” of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

2. Data Preprocessing: To explore and analysis the data set. Firstly read the data set. Then analyse the data set using the describe method and check if any null values are present our data set using “isnull()” function. After the testing the isnull function the dataset no longer contains null values. The dataset is normalized to a standardized form.

3. Data Transformation: Convert all categorical values (text values) to numerical values.In this step converting all categorical values to the numeric. This is done using the label encoder function

4. Splitting data :75 percentage of the data for training and 25 percentage of the data for testing. The splitting is done-using the train testsplit() function .

5. Model Training : After the splitting the dataset, it was applied to the datamodel. Here, to detect the network intrusion using Random Forest() model

6. Model Evaluation and Testing: The confusion Matrix, R square and classification report are used see the accuracy of the model.A confusion matrix is a table that is often used to describe the performance of a classification model (or ”classifier”) on a set of test data for which the true values are known. A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of the trained classification model.

3.3 ALGORITHM

LSTM is very good in processing sequence and time series data due to which it is used to model user behavior. While autoencoder has the ability to be used on real valued datasets and are quick concise. LSTM auto-encoders are explicitly designed to avoid the long-term dependency problem. LSTM are designed to look at the historical data to make predictions. It processes data up to (t-lookback) to make a prediction at a given time t. It takes a 3D array as input. Once the data is ready, it is divided into the train data and test-data. Train-data is used for model training and parameter tuning, while test-data is used for model evaluation.

3.4 USER INTERFACE

UI stands for User Interface. It refers to the visual elements, controls, and interactions that allow users to interact with and control a software application, website, or system. UI encompasses the design of graphical user interfaces (GUIs), web interfaces, mobile app interfaces, and other forms of digital interfaces.

The primary goal of UI design is to create an interface that is visually appealing, user-friendly, and intuitive, enabling users to easily navigate, understand, and interact with the software or system. A well-designed UI considers the needs, preferences, and expectations of the target users and aims to provide a seamless and engaging user experience.

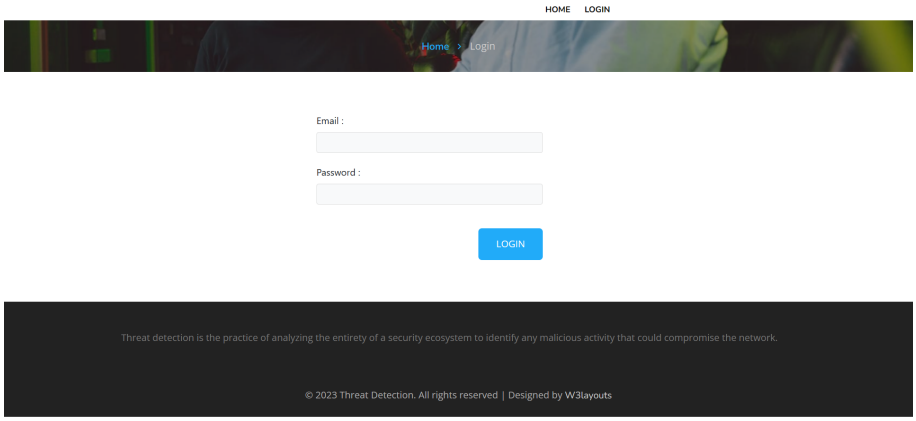


Figure 3.4: User Interface a

Figure 3.5: User Interface a

Chapter 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

1. Processor: A powerful processor is essential for handling the computational demands LSTM algorithms. A multi-core processor with a high clock speed will help improve the performance and speed of the system. Here we used intel i5 gen 11 for the implementation.

2. Memory (RAM): Sufficient memory is required to load and process images efficiently. The amount of RAM you need depends on the size and complexity of the images you plan to process. For image enhancement tasks, it is recommended to have a minimum of 8 GB of RAM, but higher amounts, such as 16 GB or more, will be beneficial for processing larger images or handling multiple requests concurrently. our system contains 8GB RAM.

3. Storage: Adequate storage space is necessary to store the images and any associated data or databases. The amount of storage you need will depend on the anticipated number and size of the images you expect to handle. Additionally, if you plan to store user-uploaded images or processed images, you will need enough storage capacity to accommodate them.Our system has 1TB HDD and 256 SSD.

4.2 SOFTWARE REQUIREMENT

For the implementation of our project we choose visual studio code as our IDE.

4.2.1 WHY VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a lightweight, free source code editor developed by Microsoft. It has gained immense popularity among developers due to its extensive features, flexibility, and wide range of extensions. Here's an overview of Visual Studio Code:

1. Cross-platform and Lightweight: VS Code is available for Windows, macOS, and Linux, ensuring compatibility across different operating systems. It is designed to be lightweight, providing fast performance while consuming minimal system resources.

2. Intuitive User Interface: VS Code has a clean and intuitive user interface that maximizes the coding space. It offers a customizable layout, allowing users to adjust the view and position of panels, sidebars, and editors according to their preferences. The user interface is designed to enhance productivity and provide a distraction-free coding environment.

3. Extensive Language Support: VS Code provides support for a wide range of programming languages out of the box, including popular ones like JavaScript, Python, Java, C++, and many more. It offers syntax highlighting, code snippets, and intelligent code completion, making it easier to write code and catch errors quickly.

4. Integrated Terminal: VS Code includes an integrated terminal that allows developers to run commands, execute scripts, and interact with the command-line interface without leaving the editor. The terminal can be customized and supports multiple instances, making it convenient to work with different environments or tools.

5. Git Integration: VS Code has excellent Git integration, enabling developers to perform version control tasks within the editor. It provides features like Git staging, commit history, branch management, and conflict resolution. Users can visually track changes, compare file revisions, and collaborate with teammates using version control.

6. Powerful Editing Features: VS Code offers a rich set of editing features to enhance productivity. These include intelligent code completion, code refactoring, formatting, find and replace, multiple cursors, code folding, and more. It supports

customizable keyboard shortcuts, allowing users to optimize their workflow and increase efficiency.

7. Integrated Debugging: VS Code provides integrated debugging capabilities for various programming languages. It allows developers to set breakpoints, step through code, inspect variables, and view call stacks, providing an efficient debugging experience. Debugging configurations can be customized to match specific project requirements.

8. Extensibility with Extensions: VS Code has a vibrant ecosystem of extensions contributed by the community. These extensions enhance the editor's functionality by adding support for additional languages, frameworks, and tools. They provide features like linters, code formatters, language servers, live server preview, and integrations with cloud platforms and build systems.

9. Integrated Package Manager: VS Code includes a built-in package manager called the Extension Marketplace. It allows users to discover, install, and manage extensions directly from within the editor. The marketplace offers a wide range of extensions to customize and extend the functionality of VS Code.

10. Live Collaboration: VS Code supports live collaboration through extensions like Live Share. It enables developers to collaborate in real-time, share their coding sessions, and work together on the same codebase, irrespective of their physical location. This feature facilitates pair programming, code reviews, and remote collaboration.

Visual Studio Code provides a powerful, user-friendly, and customizable environment for developers. Its extensive feature set, flexibility, and strong community support make it a popular choice for a wide range of programming tasks, from small scripts to large-scale projects.

Chapter 5

SOFTWARE DESCRIPTION

5.1 LANGUAGES USED

5.1.1 PYTHON

The code to implement this project is done with the language Python. Python is a popular high-level programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and allows programmers to express concepts in fewer lines of code compared to other programming languages. It has gained widespread adoption in various domains, including web development, data analysis, scientific computing, artificial intelligence, and automation.

Here are some key features and aspects of Python:

1. Syntax and Readability: Python's syntax is designed to be intuitive and easy to read. It uses indentation to define blocks of code, making it visually distinct. This feature encourages developers to write clean and readable code.

2. Interpreted Language: Python is an interpreted language, which means it does not require a compilation step before execution. The interpreter reads and executes the code line by line, allowing for rapid development and testing.

3. Multi-paradigm: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the best approach for their project.

4. Large Standard Library: Python comes with a vast standard library that provides numerous modules and packages for common tasks such as file I/O, networking, database access, and more. The standard library reduces the need for external dependencies and makes it easier to get started with various programming tasks.

5. Third-Party Libraries: Python has a rich ecosystem of third-party libraries and frameworks that extend its capabilities. These libraries cover a wide range of domains, including web development (e.g., Django, Flask), data analysis (e.g., NumPy, pandas), scientific computing (e.g., SciPy), machine learning (e.g., TensorFlow, PyTorch), and more. This extensive collection of libraries makes Python suitable for various applications.

6. Platform Independence: Python is a cross-platform language, meaning it can run on different operating systems such as Windows, macOS, and various Linux distributions. This portability makes it easier to develop and deploy applications across different environments.

7. Integration and Extensibility: Python can easily integrate with other languages like C, C++, and Java, allowing developers to leverage existing code and libraries. It offers interfaces and tools for extending Python with native code for performance-critical tasks.

8. Community and Support: Python has a large and active community of developers who contribute to its growth and provide support through online forums, documentation, and open-source projects. This vibrant community ensures that Python remains up to date and well-supported.

Python's versatility and ease of use have made it a popular choice for beginners learning to program, as well as for experienced developers working on complex projects. Its simplicity and expressiveness make it an excellent language for rapid prototyping, scripting, and building scalable applications.

PYTHON FLASK-FRAMEWORK

Flask is a lightweight web framework for Python that allows you to build web applications quickly and with minimal boilerplate code. It was developed by Armin Ronacher

and released in 2010. Flask is known for its simplicity, flexibility, and ease of use, making it a popular choice for web development.

Here are some key features and aspects of Flask:

1. Microframework: Flask is considered a microframework because it focuses on simplicity and minimalism. It provides the core functionality needed for web development, such as routing, request handling, and template rendering, without imposing strict architectural patterns or adding unnecessary layers of abstraction. This lightweight approach gives developers more freedom to structure their applications according to their preferences.

2. Routing and View Functions: Flask uses a decorator-based syntax to define routes and associate them with view functions. A route is a URL pattern that maps to a specific function. When a user accesses a particular URL, Flask invokes the associated view function to generate the response. This routing mechanism allows you to define different routes for various parts of your application and handle different HTTP methods (GET, POST, etc.) on those routes.

3. Template Engine: Flask includes a template engine called Jinja2, which enables you to generate HTML dynamically. Jinja2 allows you to separate the presentation logic from the application's business logic by defining templates with placeholders (variables) and control structures (loops, conditionals). You can pass data from your views to templates to render dynamic content.

4. HTTP Request and Response Handling: Flask provides an easy-to-use API for handling HTTP requests and generating responses. You can access request data, such as form input, query parameters, and headers, using Flask's request object. Likewise, you can generate responses with different content types (HTML, JSON, etc.) using Flask's response object or by returning values directly from view functions.

5. Extensions and Modularity: Flask follows a "micro but expandable" philosophy. While it provides a minimal core, you can extend its functionality using various Flask extensions. These extensions add extra features like database integration, authentication, form validation, and more. Flask extensions are developed by the community and can be easily integrated into your Flask applications.

6. Development Server and Debugging: Flask includes a built-in development server that simplifies the testing and debugging process. It automatically reloads the server when you modify your code, allowing you to see the changes immediately. Additionally, Flask provides a robust error handling mechanism that displays detailed error messages and a debugger to help identify and fix issues during development.

7. Integration with Other Technologies: Flask seamlessly integrates with other Python libraries and tools, making it a versatile choice for web development. You can easily use Flask with popular database systems like SQLite, MySQL, or PostgreSQL, and integrate it with ORMs (Object-Relational Mappers) such as SQLAlchemy. Flask can also work well with front-end frameworks like React, Angular, or Vue.js to build single-page applications (SPAs).

Flask's simplicity and flexibility make it a great option for small to medium-sized web applications, RESTful APIs, prototypes, and personal projects. It provides a solid foundation for web development in Python and allows developers to focus on their application's specific requirements without unnecessary complexity.

WHY PYTHON?

Python is a popular programming language that offers several advantages for implementing Insider threat detection techniques. Here are some reasons why Python is commonly used in this context:

1. Rich Ecosystem of Libraries: Python has a vast ecosystem of libraries and frameworks that provide extensive support for image processing, deep learning, and computer vision tasks. Libraries such as TensorFlow, PyTorch, and Keras offer powerful tools and APIs for building and training deep learning models like MirNet. These libraries provide pre-built functions, optimization algorithms, and utilities that simplify the implementation of complex image enhancement techniques.

2. Ease of Use and Readability: Python is known for its simplicity and readability, which makes it easier for researchers and developers to understand and modify existing code or develop new algorithms. Its clean and concise syntax allows for

efficient prototyping and experimentation, enabling quick iteration and refinement of image enhancement techniques.

3. Extensive Community Support: Python has a large and active community of developers and researchers working on various image processing and computer vision projects. This vibrant community provides support through forums, online resources, and open-source contributions. The availability of resources, tutorials, and code examples makes it easier to learn and apply advanced techniques like MirNet for image enhancement.

4. Integration with Other Technologies: Python is well-suited for integrating with other technologies and frameworks commonly used in image enhancement workflows. It seamlessly integrates with popular software libraries like OpenCV, which provides a wide range of image processing and computer vision functions. Python also facilitates interoperability with data processing tools, visualization libraries, and web frameworks, enabling end-to-end image enhancement pipelines.

5. Prototyping and Experimentation: Python's interactive and dynamic nature allows for rapid prototyping and experimentation. With Python, it is relatively straightforward to load, preprocess, and visualize images, facilitating the exploration and analysis of image enhancement techniques. This flexibility and interactivity make Python an ideal choice for researchers and developers who need to iterate quickly and evaluate the effectiveness of different approaches.

6. Platform Compatibility: Python is a cross-platform language, meaning that code developed on one operating system can be easily executed on different platforms with minimal modifications. This flexibility is beneficial when deploying MirNet-based image enhancement techniques on various devices or systems, including desktops, servers, or embedded devices.

Overall, Python's rich ecosystem, ease of use, extensive community support, integration capabilities, prototyping advantages, and cross-platform compatibility make it a popular choice for implementing MirNet image enhancement techniques. It empowers researchers and developers to effectively leverage deep learning and image processing li-

braries, enabling the development of advanced and efficient algorithms for enhancing dark images.

5.1.2 HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating the structure and content of web pages. It defines the elements and tags that structure the information displayed in a browser. HTML works alongside CSS and JavaScript to build interactive and visually appealing websites. Here's an overview of HTML:

1. Document Structure: HTML documents are structured using a hierarchical format called the Document Object Model (DOM). The DOM represents elements as nodes in a tree-like structure. An HTML document starts with the ¡html¿ tag and contains a head section (¡head¿) for metadata and a body section (¡body¿) for the visible content of the page.

2. Elements and Tags: HTML elements are the building blocks of a web page. They are defined using tags, enclosed in angle brackets (¡¿). Tags are composed of an opening tag (¡tag¿) and a closing tag (¡/tag¿), where the content resides. Some elements, known as empty elements, do not require a closing tag and are self-closing (¡tag /¿). Examples of common HTML elements include headings (¡h1¿, ¡h2¿, etc.), paragraphs (¡p¿), links (¡a¿), images (¡img¿), lists (¡ul¿, ¡ol¿, ¡li¿), and more.

3. Attributes: HTML elements can have attributes, which provide additional information about an element. Attributes are specified within the opening tag and consist of a name and a value (name="value"). Attributes define properties like the source of an image, the target of a link, the size of an input field, and more. Examples of attributes include src, href, class, id, alt, style, and data-* attributes.

4. Semantic Elements: HTML5 introduced a set of semantic elements that convey the meaning and structure of the content more accurately. These elements include ¡header¿, ¡nav¿, ¡section¿, ¡article¿, ¡aside¿, ¡footer¿, and more. Semantic elements provide better accessibility, improve search engine optimization (SEO), and help developers structure web pages in a more meaningful way.

5. Hyperlinks: HTML allows the creation of hyperlinks using the ¡a¿ (anchor) element. The href attribute specifies the URL to which the link points. Hyperlinks can be used to navigate within the same page, link to other pages, or link to external resources like images, documents, or videos. Additionally, HTML provides the ability to open links in a new window or tab using the target attribute.

6. Lists: HTML supports ordered (¡ol¿) and unordered (¡ul¿) lists. List items are defined using the ¡li¿ element, which is placed inside either an ¡ol¿ or ¡ul¿ element. Ordered lists display items with a numbering sequence, while unordered lists display items with bullet points. Nested lists can also be created by placing lists within lists.

7. Forms: HTML provides form elements (¡form¿, ¡input¿, ¡textarea¿, etc.) for capturing user input. Forms allow users to submit data to a server for processing. Various input types (text, email, password, checkbox, radio, etc.) and attributes (required, placeholder, disabled, etc.) can be used to define the behavior and appearance of form elements.

8. Multimedia: HTML supports the inclusion of multimedia content on web pages. The ¡img¿ element is used to display images, while the ¡audio¿ and ¡video¿ elements are used to embed audio and video files, respectively. These elements support different attributes for controlling playback, size, and alternative content.

9. Tables: HTML tables (¡table¿, ¡tr¿, ¡td¿) are used to organize data into rows and columns. Tables can be styled using CSS to enhance their appearance and improve readability. Additionally, HTML provides elements like ¡th¿ for table headers and attributes like colspan and rowspan for merging cells.

HTML is continually evolving, with new versions introducing additional features and elements. It provides the foundation for web development, allowing developers to structure and present content in a standardized manner across different browsers and devices.

5.1.3 CSS

CSS (Cascading Style Sheets) is a fundamental technology used for styling and formatting web documents. It describes how HTML elements should be displayed on a web

page, including their layout, colors, fonts, and other visual aspects. CSS works alongside HTML and JavaScript to create visually appealing and interactive web pages. Here's an overview of CSS:

1. Separation of Style and Structure: CSS enables the separation of a web page's content (HTML structure) from its presentation (styling). This separation allows developers to define styles independently and apply them consistently across multiple web pages. It promotes clean code, easy maintenance, and the ability to update the visual appearance of a website without modifying the underlying HTML structure.

2. Selectors and Declarations: CSS uses selectors to target specific HTML elements and apply styling rules. Selectors can be based on element types (e.g., h1, p), class names (e.g., .container, .highlighted), IDs (e.g., header, sidebar), and more. Once a selector is defined, you can specify one or more declarations, which consist of a property and a value. For example, you can set the color property to red, the font-size property to 16 pixels, or the background-image property to a specific image URL.

3. Style Inheritance and Specificity: CSS employs a cascading nature, where styles can be inherited from parent elements and overridden by more specific rules. This cascading behavior allows for efficient and modular style definitions. However, when multiple conflicting styles are applied to the same element, CSS uses a specificity calculation to determine which style takes precedence.

4. Layout and Positioning: CSS provides various layout and positioning options to control the arrangement of elements on a web page. These include properties like display (e.g., block, inline, flex), position (e.g., static, relative, absolute), float, margin, padding, and more. With these properties, you can create responsive designs, align elements, create columns, and control the flow of content.

5. Responsive Design: CSS plays a crucial role in creating responsive web designs that adapt to different screen sizes and devices. Using CSS media queries, developers can define different styles based on the screen width, height, or other device-specific features. This allows the website layout and appearance to adjust dynamically, providing an optimal viewing experience on desktops, tablets, and mobile devices.

6. Animations and Transitions: CSS provides support for animations and transitions, allowing developers to create engaging and interactive user experiences. With keyframes and animation properties, elements can move, change size, fade in/out, or perform other dynamic effects. Transitions enable smooth property changes over time, such as gradual color shifts or smooth transitions between layout states.

7. Selectors and Pseudo-Classes: CSS offers a wide range of selectors to target specific elements or groups of elements. Additionally, pseudo-classes provide a way to target elements based on their state or position within the document. Examples of pseudo-classes include :hover, :active, :first-child, :nth-child(), and many more. These selectors and pseudo-classes enhance the control and interactivity of web pages.

8. CSS Frameworks: To expedite web development and ensure consistent styling, developers often use CSS frameworks like Bootstrap, Foundation, or Bulma. These frameworks provide pre-designed CSS components, grids, and stylesheets that can be easily integrated into a project, simplifying the styling process and ensuring a cohesive visual design.

CSS is continuously evolving, and new features and enhancements are regularly introduced to the language. With CSS, developers have powerful tools to control the presentation and styling of web pages.

Chapter 6

MODULES

6.1 MODULE 1-ADMIN

Login

Manage Supervisor

View Report

Response/ Action based on report

The admin can login using the login credentials.He can manage the supervisors such as add delete edit etc.The admin has the privilege to view the report and do some action based on the report.

6.2 MODULE 2-SUPERVISOR

Login

Manage Staff

Generate staff report based on Behavioral based insider threat detection

Send Threat report to Admin as email

View admin response

The supervisor can login the system by credentials.He has the responsibility to manage the staff such as add, delete , edit etc.He can generate the staff report to the admin via email.He can view the response from Admin

6.3 MODULE 3-SYSTEM

DATA GATHERING

DATA PREPROCESSING

FEATURE EXTRACTION

MODEL TRAINING AND EVALUATION:

INSIDER THREAT DETECTION

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data.This makes the task more realistic.Some intrusion experts believe that most novel attacks are variants of known attacks and the ”signature” of known attacks can be sufficient to catch novel variants.The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

Chapter 7

IMPLEMENTATION AND PERFORMANCE EVALUATION

An experimental setup was established to evaluate and study the importance and usefulness of the proposed technique. The experimental environment consists of AMD A8 pro 1.9 GHz CPU,8 GB RAM, Windows 10 Home. The testing is carried out on Anaconda 2018.12, Build: py37 0 using Jupyter Notebook 5.7.4, which is a web based, interactive programming environment enabling user to run and edit human readable documents.

7.1 LSTM AUTOENCODER TRAINING

The model is trained with Epochs = 200, batchsize = 64, learning rate = 0.0001, activation function is “relu” and optimizer used is “Adam”. The dimensions used for input and output are same.

The difference between input and output sequence is calculated as loss function. The loss function is calculated as Mean Squared Error (MSE).

The input is reconstructed to the output, during the model training phase. The model is trained over normal or negatively labeled data i-e. Insider = 0. During testing phase if the reconstruction error is high, it is considered as anomalous behavior. The model is tested using both positive and negative samples. The reconstruction error for insider user is very high as compared to normal users. A threshold value is defined to segregate normal and malicious behavior. If the value is higher than threshold it is considered as “Insider” and if the value is lower, it is considered “Normal”.

7.2 EXPERIMENTAL RESULTS

The dataset used is the CMU CERT synthetic dataset r4.2 which consists of 1000 synthetic users out of which 70 are malicious insiders. The dataset contains only 0.03anomalous and 99.7problem of class imbalance, the technique of random over sampling is used. In which copies of anomalous instances are diffused within the dataset. The dataset is divided into train, valid, and test data for model training, validation and testing. The proportion in which data is divided includes: 70 percentage training data, 10 percentage validation data and 20 percentage testing data. Testing data contains instances of both normal and malicious data. Total number of logs are 32,770,220 out of which 17,193 are trainable parameters.

During the training phase LSTM autoencoder reconstructs input sequence to the output sequence and a loss function as MSE is calculated to identify the difference. A threshold value is set to segregate insider and normal users. If the reconstruction error is higher than threshold than it is considered “Insider”, and if lower than the threshold than it is considered “Normal”. The reconstruction error for normal user is low because the model is trained with normal data. Once the model is trained, it is then tested on mix data samples including both normal and malicious instances. The reconstruction error for insider user is very high as compared to normal users. A confusion matrix is generated which evaluates the performance of our classification model. Each row represents a class: Normal and Insider, while each column cell shows the predicted values i-e. True Positives, False Positives, True Negative and False Negative.

It helps us to identify classification accuracy along with other performance evaluation scales (Precision, F-Score, Recall etc.) The performance in terms of accuracy, precision and F1Score is calculated as shown in Figure 10. The model achieves a remarkable accuracy = 90.60 percentage, precision = 97 percentage, F1Score = 94 percentage and FPR = 9 percentage.

7.3 COMPARISON OF RESULTS WITH OTHER TECHNIQUES

Performance of our proposed algorithm is compared to other well-known techniques i-e. LSTM-CNN, LSTM-RNN, One Class SVM, Multi State LSTM CNN and Isolation forest in terms of performance evaluation metrics, dataset upon which evaluation is performed and feature set used. LSTM CNN used user activity based feature set. Detail about feature set used in LSTM-RNN is missing, One-Class SVM used domain based features, Multi State LSTM CNN used user behavior based features, Isolation Forest used psychometric observations web access patterns and our proposed approach used user session based rich feature set i-e. LogonLogoff Events,UserId, User role, Functional unit, Department, Day, Time,PC. The parameters which has a significant impact on results include: feature set, insider scenarios and the dataset used. r4.2 has multiple instances of each scenario which fully evaluate the performance of the algorithms. Upon comparison with other techniques it is observed that our novel approach produces relatively good accuracy (90.6 Percentage), precision (97 Percentage) and F1 Score (94.4 Percentage) .

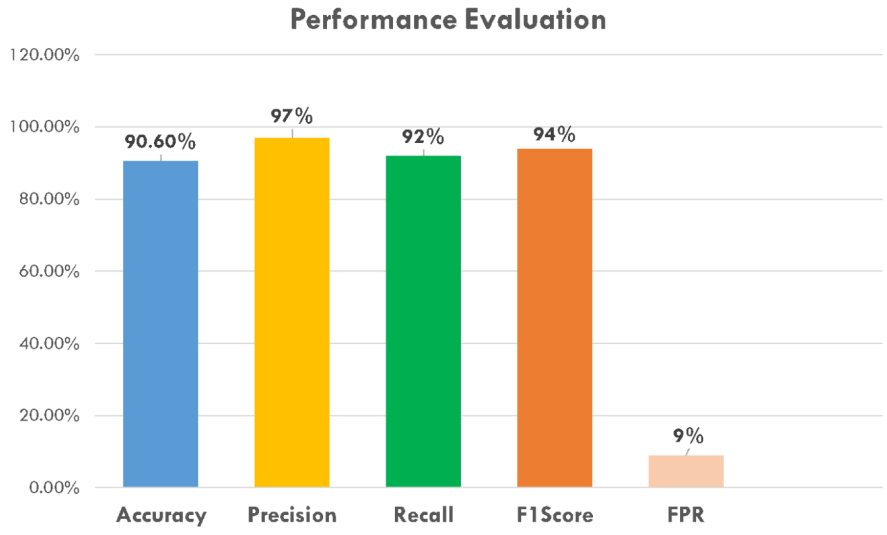


Figure 7.1: Experimental results

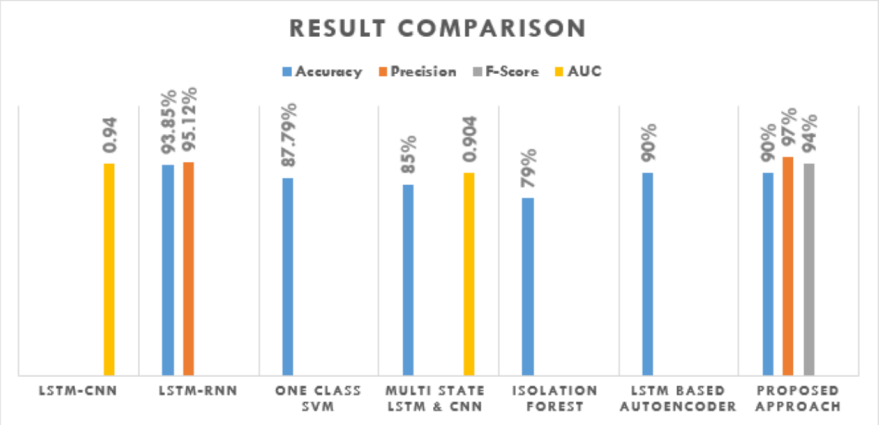


Figure 7.2: Result comparison

Chapter 8

ADVANTAGES

1. It Can Help Identify Suspicious Behavior:Before an insider threat is recognized, you need to be able to identify suspicious behavior, so that you can put a stop to the breach before it happens. Being able to recognize a possible threat to your internal database is essential so that you can begin damage control before things get worse.An insider threat program can be used by your computer staff to recognize what suspicious behavior looks like, and work out what it could lead to. From fraud to data theft and even misuse of business assets, there are many forms of internal threats that can be prevented with this type of technology.

2. Manage and Look Out for Cyber Threats:The best type of insider threat detection program will be able to not only identify suspicious behavior before it turns into something more serious but report, score, and monitor as well. This way, the staff that has been assigned to detect, watch, and manage specific types of internal threats can do so with efficiency.Insider threat programs can continuously score behavior based on past incidents, as well as what's happening currently. It can also prioritize what behavior it monitors, and what it deems lower risk. Each category of behavior can be assessed and scored based on its risk level.

3. Figure Out High-Risk Threats and Profiles:If you work within a large organization, you'll understand that it's virtually impossible for the manager or owner to keep track of every single employee, to work out if they are carrying out high-risk behavior in association with the database or not.This is where an insider threat program comes in. It can not only keep track of every employee who has access to the database, but it can also monitor each one automatically so that you can be alerted

of potential dangers in real-time, without having to wait for the report. Through risky patterns that are created, it can work out if a particular behavior is being repeated, and whether it poses a risk worth mentioning.The reality of insider threats is that they often follow the same patterns. This is why using an insider threat program is inherent to the security of your business's database. Once you are aware of the patterns that are cause for concern, you can respond to the threats a lot quicker, and save and protect more of your data.

Chapter 9

CONCLUSION AND FUTURE ENHANCEMENTS

Studied the insider threat problem, and identified that mitigating this problem is a challenging task. Now a days, mitigation against insider threat is achieved by implementing user access controls, user behavior monitoring and physical security controls. In this work a Deep Learning based insider attack detection scheme is presented. The main aim behind the development of this scheme is its application on user technical data within an organization with low processing and memory requirements. Moreover, the developed system is simple and adaptable with minimum domain knowledge requirement

Different insider threat scenarios are used and "LSTM-Autoencoder" is used for insider threat detection. Model is trained and tested on CMU CERT V4.2. The platform used for evaluation of the scheme is Anaconda 2018.12, Build: py370 using Jupyter Notebook 5.7.4. Moreover the performance of the proposed algorithm has been compared to other well-known techniques i.e. LSTM-CNN, Random Forest, LSTM-RNN, One Class SVM, Markov Chain Model, Multi State LSTM CNN, Gated Recurrent Unit Skipgram. The comparison showed that our novel approach produces relatively good accuracy(90.60 percentage), precision(97 percentage) and F1 Score (94 percentage).

In order to create a robust Insider detection system, we need to create more diverse insider threat scenarios, as there is a lack of publicly available threat scenarios. This will help us in solving the insider problems with more creativity, high quality and accuracy.

Bibliography

[1] Insider Report 2018, CA Technol., New York, NY, USA, 2018.

[2] Insider Threat Report 2019, CA Technol., San Jose, CA, USA, 2019.

[3] S. R. Band, D. M. Cappelli, L. F. Fischer, A. P. Moore, E. D. Shaw, and R. F. Trzeciak, "Comparing insider IT sabotage and espionage: A modelbased analysis," Carnegie Mellon Univ., Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2006TR-026, 2006.

[4] P. Chattopadhyay, L. Wang, and Y.-P. Tan, "Scenario-based insider threat detection from cyber activities," IEEE Trans. Comput. Social Syst., vol. 5, no. 3, pp. 660–675, Sep. 2018.

[5] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in Proc. Int. Conf. Comput. Sci. Cham, Switzerland: Springer, 2018.

[6] W. Jiang, Y. Tian, W. Liu, and W. Liu, "An insider threat detection method based on user behavior analysis," in Proc. Int. Conf. Intell. Inf. Process. Amsterdam, The Netherlands: International Federation for Information Processing, 2018, pp. 421–429.

[7] C. Liu, Y. Zhong, and Y. Wang, "Improved detection of user malicious behavior through log mining based on IHMM," in Proc. 5th Int. Conf. Syst. Informat. (ICSAI), Nov. 2018, pp. 1193–1198.

[8] Z. Zamanian, A. Feizollah, N. B. Anuar, L. B. M. Kiah, K. Srikanth, and S. Kumar, "User profiling in anomaly detection of authorization logs," in Computational Science and Technology. Singapore: Springer. 2019.

[9] J. Jiang, J. Chen, K.-K.-R. Choo, K. Liu, C. Liu, M. Yu, and P. Mohapatra, "Prediction and detection of malicious insiders' motivation based on sentiment profile on webpages and emails," in Proc. MILCOM, Oct. 2018, pp. 1–6.

[10] D. Zhang, Y. Zheng, Y. Wen, Y. Xu, J. Wang, Y. Yu, and D. Meng, "Rolebased log analysis applying deep learning for insider threat detection," in Proc. SecArch, Toronto, ON, Canada, Jan. 2018, pp. 18–20.

[11] K. A. Tabash and J. Happa, "Insider-threat detection using Gaussian mixture models and sensitivity profiles," Comput. Secur., vol. 77, pp. 838–859, Aug. 2018.

[12] O. Lo, W. J. Buchanan, P. Griffiths, and R. Macfarlane, "Distance measurement methods for improved insider threat detection," Secur. Commun. Netw., vol. 2018, pp. 1–18, Jan. 2018.

[13] A. Gamachchi, L. Sun, and S. Boztas, "Graph based framework for malicious insider threat detection," in Proc. 50th Hawaii Int. Conf. Syst. Sci. (HICSS), 2017, p. 10.

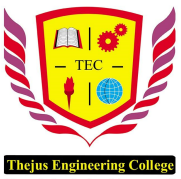
[14] F. Meng, F. Lou, Y. Fu, and Z. Tian, "Deep learning based attribute classification insider threat detection for data security," in Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace, Jun. 2018, pp. 576–581.

[15] A. Shaghaghi, S. S. Kanhere, M. A. Kaafar, E. Bertino, and S. Jha, "Gargoyle: A network-based insider attack resilient framework for organizations," in Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN), Oct. 2018, pp. 553–561.

[16] D. Patil and B. Meshram, "Network packet analysis for detecting malicious insider," in Proc. 3rd Int. Conf. Converg. Technol., 2018, pp. 1–8.

[17] L. Liu, O. De Vel, C. Chen, J. Zhang, and Y. Xiang, "Anomalybased insider threat detection using deep autoencoders," in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), Nov. 2018, pp. 39–48.

[18] L. Lin, S. Zhong, C. Jia, and K. Chen, "Insider threat detection based on deep belief network feature representation," in Proc. Int. Conf. Green Informat. (ICGI), Aug. 2017, pp. 54–59.



Department of Computer Science and Engineering
Thejus Engineering College
Vellarakkad, Thrissur - 680 584
(http://www.thejusengg.ac.in)