WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Privilege Escalation Attack Detection and Mitigation in Cloud Using Machine Learning

Abhi

8th Sem, Dept. of CSE, RLJIT

1. INTRODUCTION

Cloud computing is a new way of thinking about how to facilitate and provide services through the Internet. The current infrastructure. Cloud storage providers adopt fundamental security measures for their systems and the data they handle, including encryption, access control, and authentication. Depending on the accessibility, speed, and frequency of data access, the cloud has an almost infinite capacity for storing any type of data in different cloud data storage structures. Sensitive data breaches might occur due to the volume of data that moves between businesses and cloud service providers, both inadvertent and malicious. The characteristics that make online services easy to use for workers and IT systems also make it harder for businesses to prevent unwanted access [2]. Authentication and open Interfaces are new security vulnerabilities that Cloud services subject enterprises face. Hackers with advanced skills utilize their knowledge to access Cloud systems Machine learning employs a variety of approaches and algorithms to address the security challenge and better manage data. Many datasets are private and cannot be released owing to privacy concerns, or they may be missing crucial statistical properties [3], [4].

1.1 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Platform – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1) Software : Anaconda

2) Primary Language : Python

3) Frontend Framework : Flask

4) Back-end Framework : Jupyter Notebook

5) Database : Sqlite3

6) Front-End Technologies : HTML, CSS, JavaScript and Bootstrap4

1.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture – All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory – All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage – Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter – Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1)Operating System : Windows Only

2)Processor : i5 and above

3)Ram : 4gb and above

4)Hard Disk : 50 GB

2. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement; as only minimal or null changes are required for implementing this system.

2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3. LITERATURE SURVEY

3.1 Cloud Security Threats and Solutions: A Survey:

Cloud computing is the on-demand availability of PC framework resources. Especially information storage and handling power, without direct unique administration by the customer. It has provided customers with public and private computing and data storage on a single platform across the Internet. Aside from that, it faces several security threats and issues, which may slow down the adoption of cloud computing models. Cloud computing security threats, difficulties, strategies, and solutions are discussed in this paper. Numerous people raised security concerns in a previous survey. Another survey looks at the cloud computing architectural model, and a few of them detail security challenges and techniques. This article brings together all the security concerns, difficulties, techniques, and solutions in one place.

3.2 Cloud-based email phishing attack using machine and deep learning algorithm:

Cloud computing refers to the on-demand availability of personal computer system assets, specifically data storage and processing power, without the client's input. Emails are commonly used to send and receive data for individuals or groups. Financial data, credit reports, and other sensitive data are often sent via the Internet. Phishing is a fraudster's technique used to get sensitive data from users by seeming to come from trusted sources. The sender can persuade you to give secret data by misdirecting in a phished email. The main problem is email phishing attacks while sending and receiving the email. The attacker sends spam data using email and receives your data when you open and read the email. In recent years, it has been a big problem for everyone. This paper uses different legitimate and phishing data sizes, detects new emails, and uses different features and algorithms for classification. A modified dataset is created after measuring the existing approaches. We created a feature extracted comma-separated values (CSV) file and label file, applied the support vector machine (SVM), Naive Bayes (NB), and long short-term memory (LSTM) algorithm. This experimentation considers the recognition of a phished email as a classification issue. According to the comparison and implementation, SVM, NB and LSTM performance is better and more accurate to detect email phishing attacks. The classification of email attacks using SVM, NB, and LSTM classifiers achieve the highest accuracy of 99.62%, 97% and 98%, respectively.

3.3 Cloud computing platform: Performance analysis of prominent cryptographic algorithms:

With advancements in science and technology, cloud computing is the next big thing in the industry. Cloud cryptography is a technique that uses encryption algorithms to secure data. The significant advantage of cloud storage is no difficulty to get to, diminished equipment, low protection, and fixing cost so every association is working with the cloud. Encryption is the process of encoding information to prevent unauthorized access. Nowadays, we desire to secure the information that is to be stored in our computer or transmitted utilizing the internet against attacks. The cryptographic method depends on their response time, confidentiality, bandwidth, and integrity. Furthermore, security is a significant factor in cloud computing for ensuring client data is placed on the safe mode in the cloud. Our research paper compares the efficiency, usage, and utility of available cryptography algorithms. Evaluation results suggest which algorithm is better for which type of data and environment.

3.4 Techniques and countermeasures for preventing insider threats:

With the wide use of technologies nowadays, various security issues have emerged. Public and private sectors are both spending a large portion of their budget to protect the confidentiality, integrity, and availability of their data from possible attacks. Among these attacks are insider attacks which are more serious than external attacks, as insiders are authorized users who have legitimate access to sensitive assets of an organization. As a result, several studies exist in the literature aimed to develop techniques and tools to detect and prevent various types of insider threats. This article reviews different techniques and countermeasures that are proposed to prevent insider attacks. A unified classification model is proposed to classify the insider threat prevention approaches into two categories (biometric-based and asset-based metric). The biometric-based category is also classified into (physiological, behavioral and physical), while the asset metric-based category is also classified into (host, network and combined). This classification systematizes the reviewed approaches that are validated with empirical results utilizing the grounded theory method for rigorous literature review. Additionally, the article compares and discusses significant theoretical and empirical factors that play a key role in the effectiveness of insider threat prevention approaches (e.g., datasets, feature domains, classification algorithms, evaluation metrics, real-world simulation, stability and scalability, etc.). Major challenges are also highlighted which need to be considered when deploying real-world insider threat prevention systems. Some research gaps and recommendations are also presented for future research directions.

3.5 Smart home security: challenges, issues and solutions at different IoT layers:

The Internet of Things is a rapidly evolving technology in which interconnected computing devices and sensors share data over the network to decipher different problems and deliver new services. For example, IoT is the key enabling technology for smart homes. Smart home technology provides many facilities to users like temperature monitoring, smoke detection, automatic light control, smart locks, etc. However, it also opens the door to new set of security and privacy issues, for example, the private data of users can be accessed by taking control over surveillance devices or activating false fire alarms, etc. These challenges make smart homes feeble to various types of security attacks and people are reluctant to adopt this technology due to the security issues. In this survey paper, we throw light on IoT, how IoT is growing, objects and their specifications, the layered structure of the IoT environment, and various security challenges for each layer that occur in the smart home. This paper not only presents the challenges and issues that emerge in IoT-based smart homes but also presents some solutions that would help to overcome these security challenges.

Table 3.1: Comparison Tabular Format for Literature Survey:

S.NO	TITLE & AUTHORS	METHODOLOGY	PROPOSED SYSTEM	CONS	CONCLUSION
1	**Title:** Cloud Security Threats and Solutions: A Survey **Authors:** Umer Ahmed Butt et.al.,	This paper compiles and analyzes security threats, challenges, strategies, and solutions related to cloud computing. It reviews existing surveys and studies to provide a comprehensive overview of cloud computing security concerns.	The proposed system is an overview and analysis of cloud computing security issues, challenges, and solutions. It aims to present a consolidated resource for readers to understand and address security concerns in cloud computing.	1. It doesn't mention any recent developments or emerging trends in cloud computing security, which could be relevant for readers looking for up-to-date information.	Cloud computing offers numerous benefits, but security remains a significant concern. This paper highlights the importance of addressing security threats and provides insights into strategies and solutions to enhance cloud computing security.
2	**Title:** Cloud-based email phishing attack using machine and deep learning algorithm **Authors:** Hamza Aldabbas et.al.,	This study focuses on detecting email phishing attacks by using various features and machine learning algorithms. It involves creating a modified dataset, feature extraction, and classification using SVM, Naive Bayes, and LSTM algorithms.	The proposed system aims to enhance email security by accurately detecting phishing attacks. It utilizes machine learning techniques and a modified dataset to achieve high detection accuracy.	1. It doesn't mention potential limitations or challenges faced during the study, which could provide a more comprehensive understanding of the research context.	SVM, Naive Bayes, and LSTM classifiers demonstrate effective performance in detecting email phishing attacks, with SVM achieving the highest accuracy of 99.62%. This research contributes to improved email security.
3	**Title:** Cloud computing platform: Performance analysis of prominent cryptographic algorithms **Authors:** Abdullah Ajmal et.al.,	This research evaluates various cryptographic algorithms for securing data in cloud computing. The study assesses the efficiency, usage, and suitability of different algorithms based on factors like response time, confidentiality, bandwidth, and integrity.	The proposed system aims to enhance data security in cloud computing by identifying the most effective cryptographic algorithms for different types of data and environments.	1. It doesn't mention potential limitations or challenges faced during the research, which could provide a more comprehensive understanding of the study's context.	The research findings provide valuable insights into the performance and applicability of various cryptographic algorithms in cloud computing, helping organizations make informed decisions to secure their data.
4	**Title:** Techniques and countermeasures for	This article reviews techniques and countermeasures for	A unified classification model categorizes insider threat prevention	1. While it highlights challenges and	The article offers valuable insights into insider threat prevention techniques, their

	preventing insider threats **Authors:** Rakan A. Alsowail & Taher Al-Shehari	preventing insider attacks, classifying them into biometric-based (physiological, behavioral, physical) and asset-based (host, network, combined) categories. Empirical validation is conducted using grounded theory for rigorous literature review, comparing factors affecting effectiveness and highlighting challenges.	approaches, providing a systematic overview of the field. It distinguishes between biometric and asset-based metrics and discusses various factors influencing the effectiveness of these approaches.	research gaps, it doesn't explicitly mention limitations of the review itself, which could affect the comprehensiveness of the study.	classification, and the factors impacting their effectiveness. It identifies research gaps and provides recommendations for future studies in this domain.
5	**Title:** Smart home security: challenges, issues and solutions at different IoT layers **Authors:** Haseeb Touqeer et.al.,	This survey paper provides an overview of IoT technology, its growth, object specifications, and the layered structure of the IoT environment. It delves into security challenges at each layer in the context of smart homes and offers potential solutions.	The paper highlights the rapid growth of IoT technology and its application in smart homes. It discusses security challenges, including data privacy and device control, and suggests solutions to address these issues.	1. predominantly focuses on security challenges, but it could benefit from a broader perspective that also considers other aspects of IoT-based smart homes, such as usability, interoperability, and user acceptance.	The survey emphasizes the significance of IoT in smart homes while acknowledging the security and privacy challenges it poses. It presents potential solutions to enhance the security of IoT-based smart homes.
6	**Title:** Insider Threat Detection Using An Unsupervised Learning Method: COPOD **Authors:** Xiaoshuang Sun et.al.,	This study employs a tree structure approach to analyze user behavior and generate feature sequences. It integrates the Copula Based Outlier Detection (COPOD) method to identify anomalies in these feature sequences, specifically targeting insider threat detection. The research utilizes the CERT-IT dataset for experimentation and compares its approach with established methods like Isolation Forest.	The proposed system combines tree structure analysis with the COPOD method to detect insider threats. It generates feature sequences from user behavior data, allowing for the identification of abnormal user activities. The system's effectiveness is evaluated using the CERT-IT dataset and benchmarked against conventional methods like Isolation Forest.	1. The study's reliance on a single dataset (CERT-IT) may limit the generalizability of its findings. Insider threat scenarios can vary widely across different organizations and industries. 2. While the proposed method shows promise, it may not address scalability issues for large-scale networks or organizations with extensive user activity.	Insider threats pose significant challenges to internal network security. This study presents an approach that leverages tree structure analysis and COPOD to detect abnormal user behaviors, addressing these threats. Experimental results on the CERT-IT dataset demonstrate the method's effectiveness, surpassing traditional methods such as Isolation Forest.

				The study should consider the potential challenges of implementing the approach in real-world, high-volume environments.	

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

In recent studies, researchers worked on detecting and classifying privileged elevation attacks from insider personnel. They proposed different machine learning and deep learning techniques to counter these challenges. Techniques like SVM, Naïve Bayes, CNN, Linear Regression, PCA, Random Forest, and KNN were applied in recent studies. However, the demand for fast and effective machine learning algorithms is highly valued with the diversity of attack types. Therefore an effective and efficient strategy is required to detect, classify and mitigate these insider attacks.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- Insider attacks are difficult to identify and prevent because they exist beneath the enterprise-level security defense measures and frequently have privileged access to the network.
- Detecting and classifying insider threats has become difficult and time-consuming

4.2 Proposed System:

To the best of our knowledge, this is the first paper that deals with measuring the performance of the four machine learning algorithms (Random Forest, AdaBoost, XGBoost, and LightGBM) on classifying insider attacks and using this (algorithm performance) to quickly identify appropriate defense tools that improve the level of security protection. Recent insider threat detection and classification studies used different models and ensemble techniques. Those studies individually implemented the models on different datasets and then gave the classification results. This paper implemented the four ensemble models on a single customized dataset to better detect and classify insider threats. Our study presented the best results of applied ensemble algorithms.

4.2.1 Advantages of proposed system:

1. Overall, LightGBM performed best.
2. However, some other algorithms, such as RF or AdaBoost, may perform better on some internal attacks (Behavioral Biometrics attacks) or other internal attacks.
3. Therefore, there is room for incorporating more than one machine learning algorithm to obtain a stronger classification in multiple internal attacks.
4. Among the proposed algorithms, the LightGBM algorithm provides the highest accuracy of 97%; the other accuracy values are RF at 86%, AdaBoost at 88%, and XGBoost at 88.27%.

4.3 FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

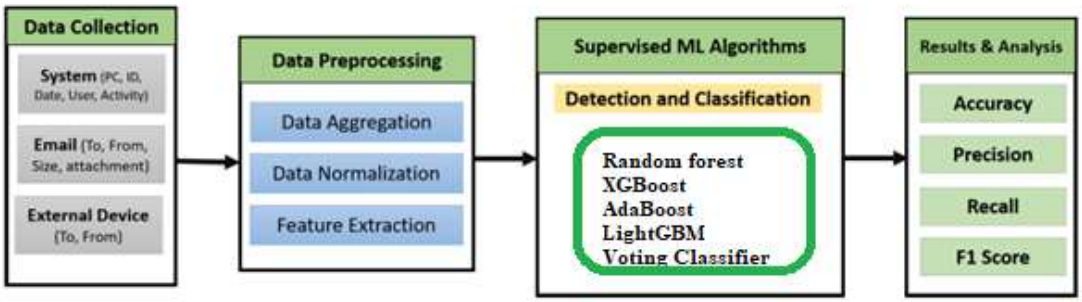


Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

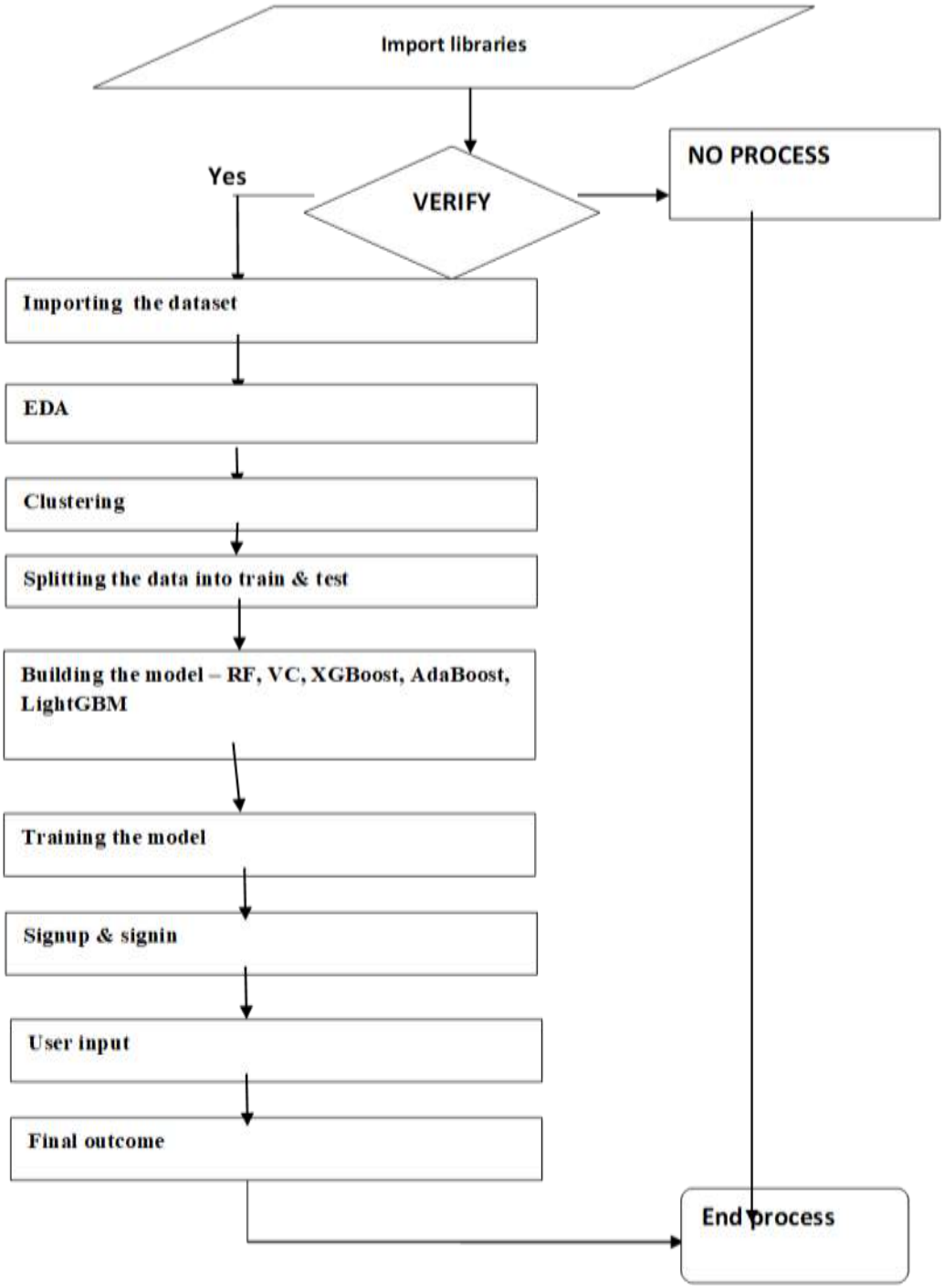


Fig.5.2 System architecture

Fig.5.2 System architecture

5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Use case diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

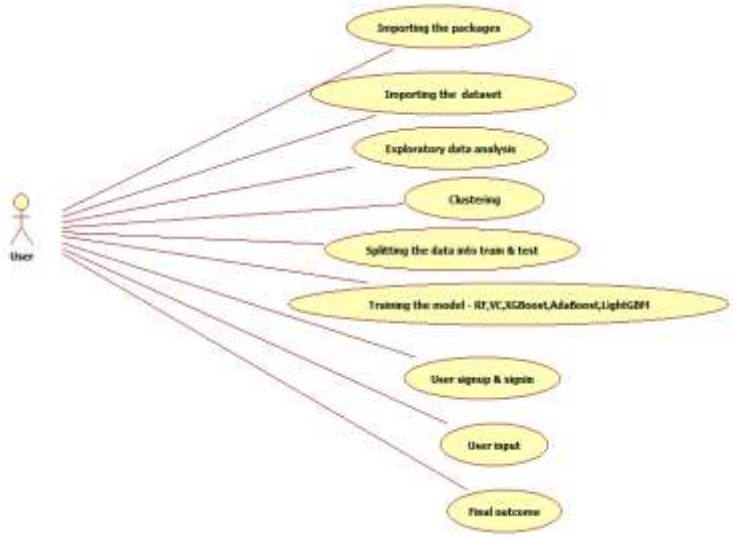


Fig.5.3 Use case diagram

Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

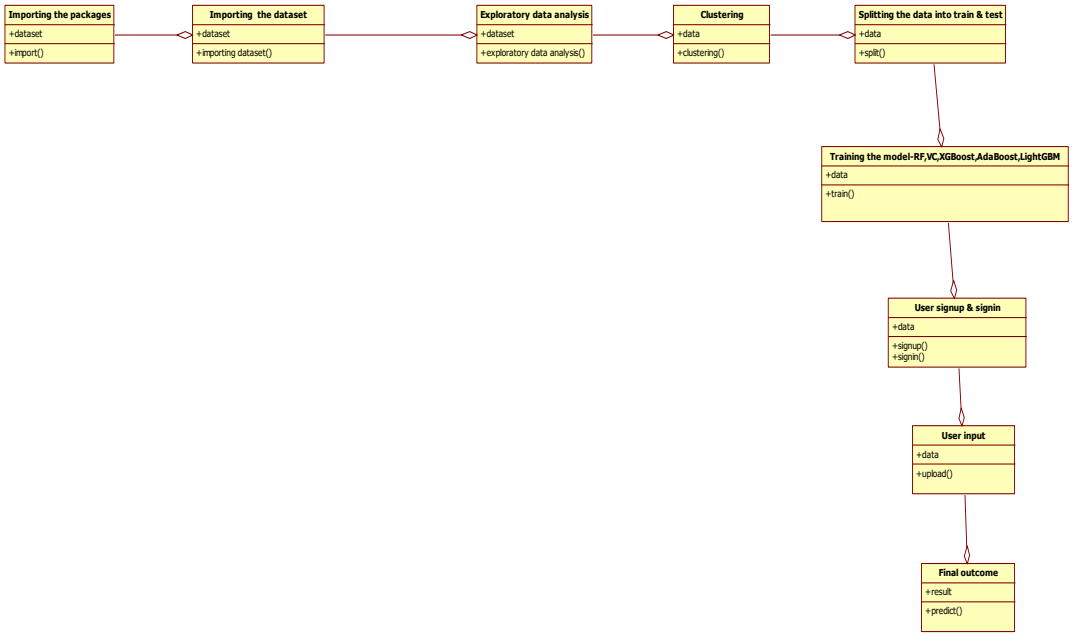


Fig.5.4 Class diagram

Activity diagram

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

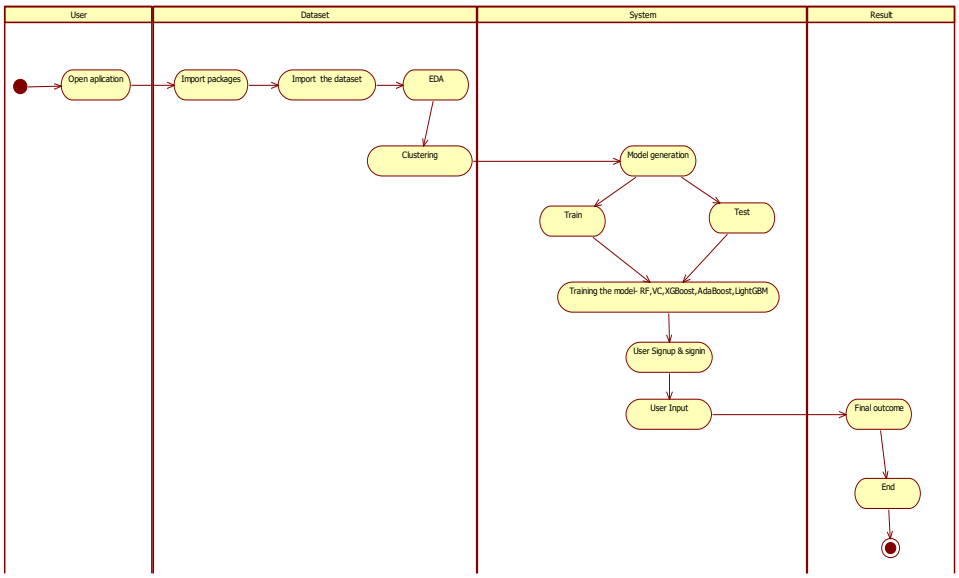


Fig.5.5 Activity diagram

Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

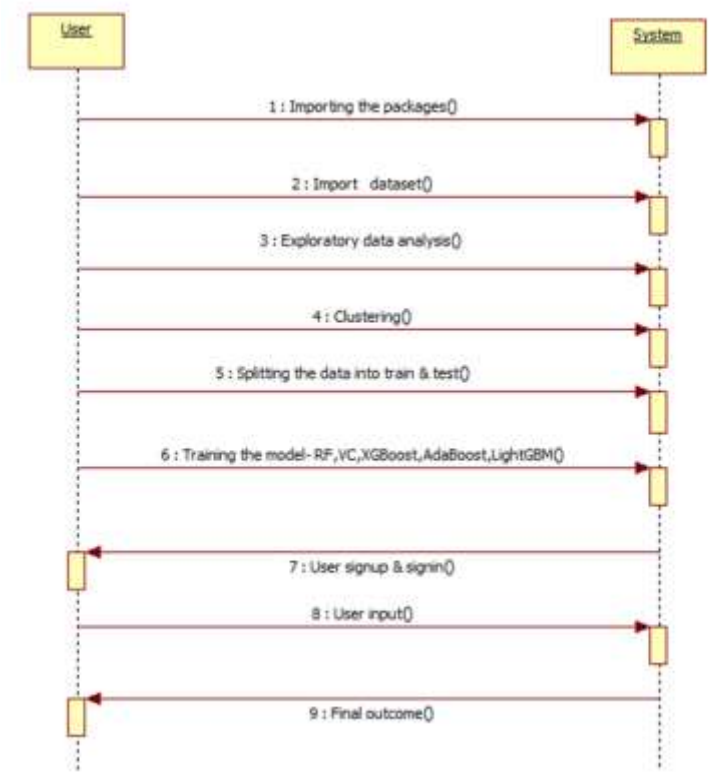


Fig.5.6 Sequence diagram

Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

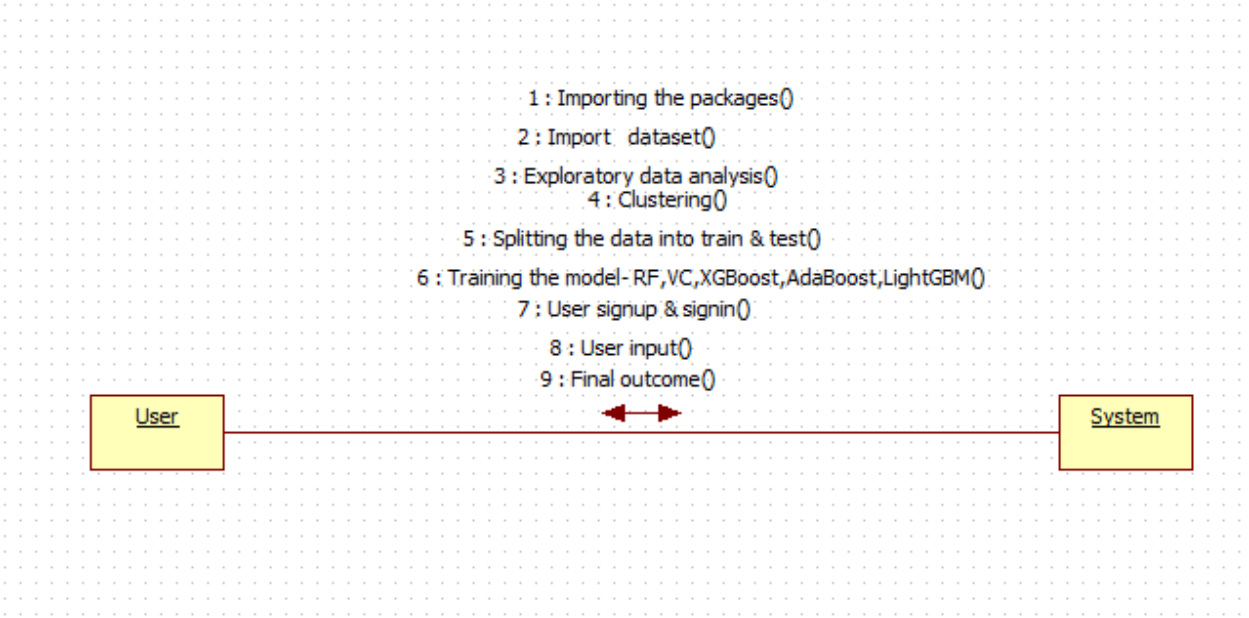


Fig.5.7 Collaboration diagram

Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

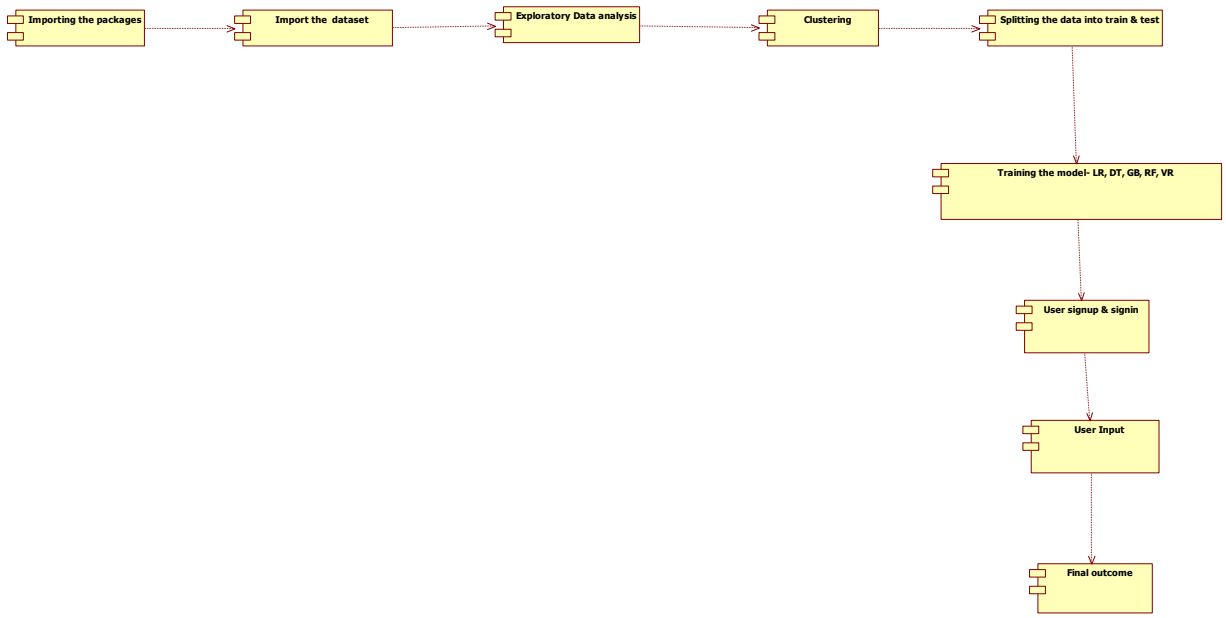


Fig.5.8 Component diagram

Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

Fig.5.9 Deployment diagram

6. IMPLEMENTATION

MODULES:

1. Importing Libraries:

Import the necessary Python libraries for data manipulation, visualization, and machine learning, such as pandas, numpy, matplotlib, seaborn, scikit-learn, LightGBM, XGBoost, etc.

2. Importing Dataset:

Obtain the dataset that contains information related to the insider activities.

Dataset Link:

Dataset Description:

The CERT dataset, maintained by the Computer Emergency Response Team (CERT) at the Software Engineering Institute, provides a collection of cybersecurity-related data for research and analysis. It includes information on vulnerabilities, incidents, and various security-related data to help researchers and professionals understand and improve cybersecurity practices and solutions.

3. EDA (Exploratory Data Analysis):

- Checking for Null Values: Examine the dataset for missing values. Decide on a strategy to handle these missing values (imputation or removal).

- Feature Selection: Identify and select features that are most relevant for detecting privilege escalation attacks. Remove irrelevant or redundant features to improve model efficiency.

- Data Visualization: Create visualizations (e.g., histograms, scatter plots, correlation matrices) to understand the data distribution and relationships between features.

- Data Cleaning: Handle any outliers or anomalies that might affect the performance of the machine learning models.

4. Clustering (KMeans):

Consider using KMeans clustering to group similar data points together. This can help in identifying patterns and potentially separating normal and abnormal user behaviors. Clustering can also be useful for creating more targeted features for the subsequent machine learning algorithms.

5. Machine Learning Algorithms:

Train and evaluate multiple machine learning algorithms for privilege escalation attack detection. It's important to consider both traditional ensemble methods and gradient boosting algorithms due to their effectiveness in handling complex patterns in data.

- Lightgbm
- Xgboost
- Adaboost
- Random Forest
- Voting classifier

Algorithms:

LightGBM: LightGBM is a gradient boosting ensemble method that is used by the Train Using AutoML tool and is based on decision trees. As with other decision tree-based methods, LightGBM can be used for both classification and regression. LightGBM is optimized for high performance with distributed systems.

XGBoost: How XGBoost Works - Amazon SageMaker XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

AdaBoost: AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

RF: Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

VC: A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

6.2 SAMPLE CODE:

```
import
panda
s as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from lifelines import KaplanMeierFitter
        class KaplanMeierUtils():
            """
            Module of reusable function and utilities for
            survival analysis.

            """
```

```
def __init__(self):
    pass

def PlotKaplanMeierEstimatesForCategoricalVariables(self, data = None, categorical_columns=[]):

    '''
    Purpose:
        Plots the Kaplan Meier Estimates For Categorical Variables in the data.
        **NOTE: The Kaplan-Meier estimator is used to estimate the survival function.
        The visual representation of this function is usually called the Kaplan-Meier
        curve, and it shows what the probability of an event (for example, survival)
        is at a certain time interval. If the sample size is large enough, the curve
        should approach the true survival function for the population under
        investigation.
    Parameters:
        1. data: the dataset.
        2. categorical_columns: all the categorical data features as a list.
    Return Value:
        NONE.
    Reference:
https://lifelines.readthedocs.io/en/latest/fitters/univariate/KaplanMeierFitter.html#lifelines.fitters.kaplan_meier_fitter.KaplanMeierFitte
r
    '''
    categoricalData = data.loc(axis=1)[categorical_columns]
    fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(18,10))
    plt.tight_layout(pad=5.0)
    def km_fits(data, curr_Feature = None):
        '''
        Purpose:
            Generates the Kaplan Meier fits for fitting the Kaplan-Meier
            estimate for the survival function.
        Parameters:
            1. data: the dataset.
            2. curr_Feature: the current feature under consideration.
        Return Value:
            kmfits: the Kaplan-Meier estimates.
        '''
        curr_feature_range = np.unique(data[curr_Feature])
```

```
        X = [data[data[curr_Feature]==x]['time'] for x in curr_feature_range]
        Y = [data[data[curr_Feature]==y]['DEATH_EVENT'] for y in curr_feature_range]
        fit_label = [str(curr_Feature + ': ' + str(feature_range_i)) for feature_range_i in curr_feature_range]
        kmfits = [KaplanMeierFitter().fit(durations = x_i,
                                          event_observed = y_i,
                                          label = fit_label[i]) for i,(x_i, y_i) in enumerate(zip(X,Y))]
        return kmfits
    for idx, feature in enumerate(categorical_columns):
        cat_fits = km_fits(data = data,
                           curr_Feature = feature)
        [x.plot(title=feature, ylabel="Survival Probability", xlabel="Days",
                ylim=(0,1.1), xlim=(0,300),
                ci_alpha=0.1, ax=ax.flatten()[idx]) for x in cat_fits]
    ax.flatten()[-1].set_visible(False)
    fig.suptitle("Kaplan Meier Estimates for Categorical Variables ", fontsize=16.0)
    plt.show()

    return None

def PlotKaplanMeierEstimatesForContinuousVariables(self, data = None, continuous_columns=[]):

    '''
    Purpose:
        Plots the Kaplan Meier Estimates For Continuous Variables in the data.
        **NOTE: The Kaplan-Meier estimator is used to estimate the survival function.
        The visual representation of this function is usually called the Kaplan-Meier
        curve, and it shows what the probability of an event (for example, survival)
        is at a certain time interval. If the sample size is large enough, the curve
        should approach the true survival function for the population under
        investigation.
    Parameters:
        1. data: the dataset.
        2. continuous_columns: all the continuous data features as a list.
    Return Value:
        NONE.
    Reference:
```

https://lifelines.readthedocs.io/en/latest/fitters/univariate/KaplanMeierFitter.html#lifelines.fitters.kaplan_meier_fitter.KaplanMeierFitter

```
'''
continuousData = data.loc(axis=1)[continuous_columns]
fig, ax = plt.subplots(nrows=3, ncols=3, figsize=(18,15))
plt.tight_layout(pad=5.0)
def km_fits(data, curr_Feature = None, split_points = None):
    '''
    Purpose:
        Generates the Kaplan Meier fits for fitting the Kaplan-Meier
        estimate for the survival function.
    Parameters:
        1. data: the dataset.
        2. curr_Feature: the current feature under consideration.
        3. split_points: the data split points to cut the data.
    Return Value:
        kmfits: the Kaplan-Meier estimates.
    '''
    bins = pd.cut(x=data[curr_Feature],bins=split_points)
    curr_feature_range = np.unique(bins)
    curr_feature_group = str(curr_Feature) + "_group"
    data[curr_feature_group] = pd.cut(x=data[curr_Feature], bins=split_points)
    X = [data[data[curr_feature_group] == bin_range]['time'] for bin_range in curr_feature_range]
    Y = [data[data[curr_feature_group] == bin_range]['DEATH_EVENT'] for bin_range in curr_feature_range]
    fit_label = [str(str(feature_range_i).replace(',',' -').replace(']',')')) for feature_range_i in curr_feature_range]
    data.drop(curr_feature_group, axis=1, inplace=True)
    kmfits = [KaplanMeierFitter().fit(durations = x_i,
                                      event_observed = y_i,
                                      label=fit_label[i]) for i,(x_i, y_i) in enumerate(zip(X,Y))]
    return kmfits
data_split_points = [[39.0,60.0,80.0,100.0],3,[0,30.0,45.0,100.0],3,3,3,3]
for idx, feature in enumerate(continuous_columns):
    cont_fits = km_fits(data = data,
                        curr_Feature = feature,
                        split_points = data_split_points[idx])
    [x.plot(title=feature, ylabel="Survival Probability", xlabel="Days",
            ylim=(0,1.1), xlim=(0,300), ci_alpha=0.1,
            ax=ax.flatten()[idx]) for x in cont_fits]
ax.flatten()[-1].set_visible(False)
ax.flatten()[-2].set_visible(False)
```

```
fig.suptitle("Kaplan Meier Estimates for Continuous Variables ", fontsize=16.0, y=1.0)
plt.show()
```

7. SYSTEM TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

8.1Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.

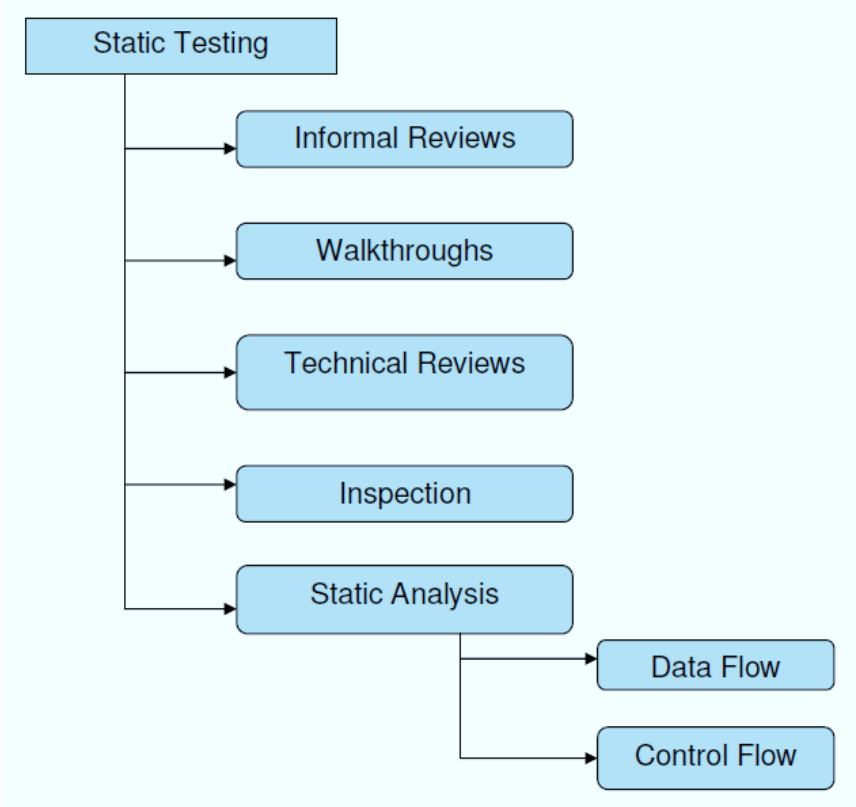


Fig.7.1 Structural Testing

Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).

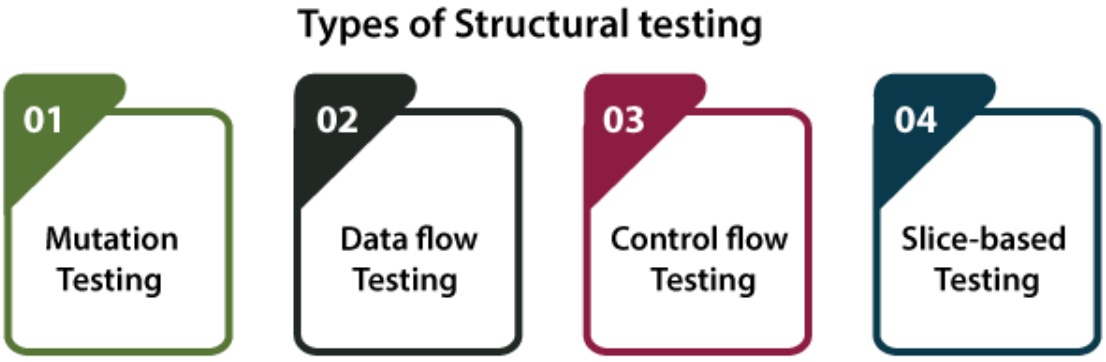
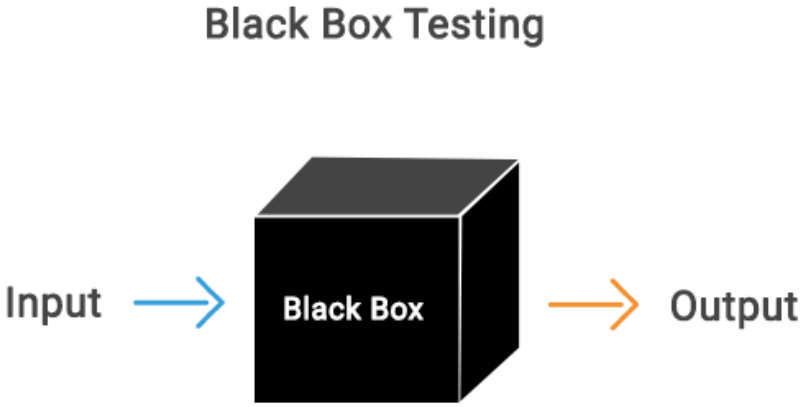


Fig.7.2 Behavioral Testing

Behavioral Testing:

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



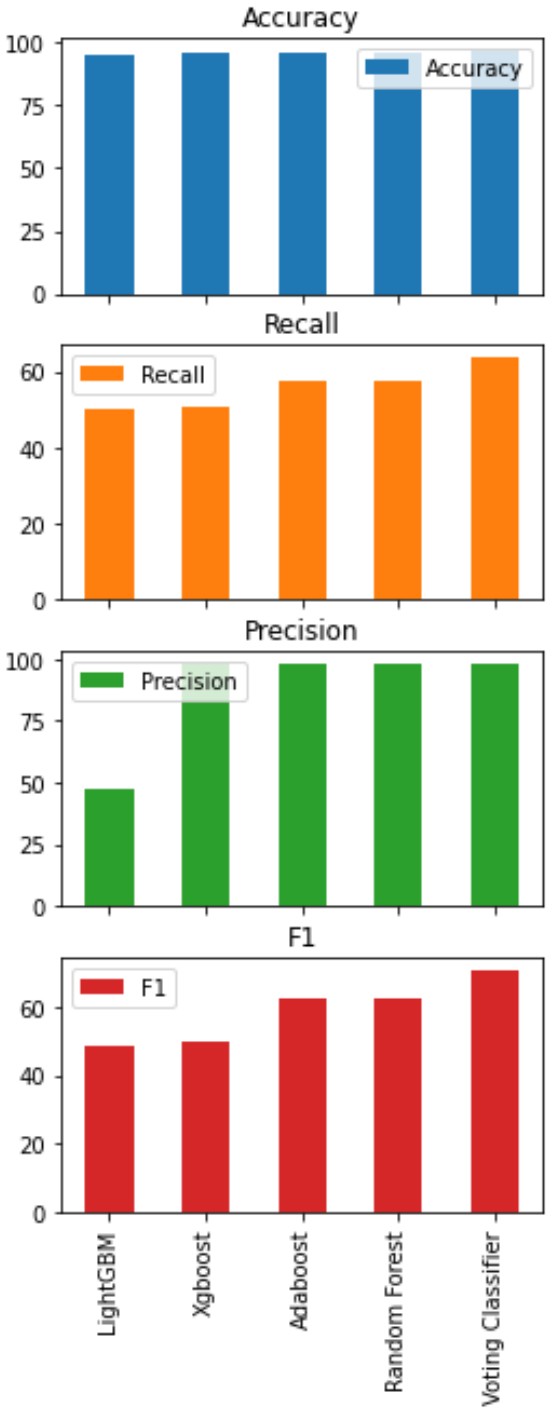
8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User signin	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

8. SCREENSHOTS

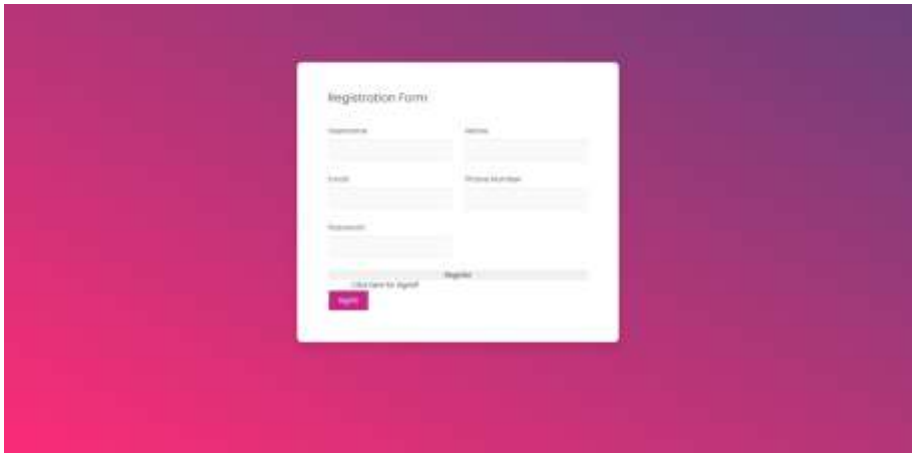
SCREENS:

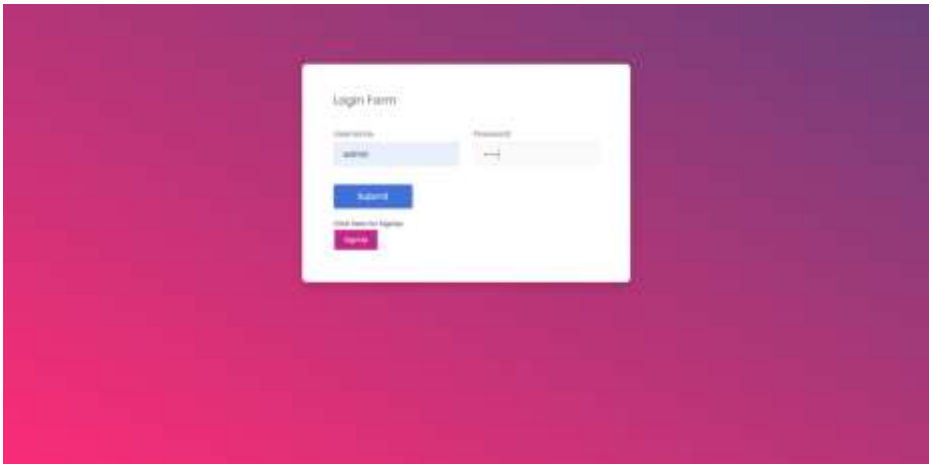
Comparison



Screenshots


PRIVILEGE ESCALATION ATTACK
DETECTION AND MITIGATION IN
CLOUD USING MACHINE
LEARNING


Registration Form


Login Form

CONCLUSION

The malicious insider becomes a crucial threat to the organization since they have more access and opportunity to produce significant damage. Unlike outsiders, insiders possess privileged and proper access to information and resources. This paper proposed machine learning algorithms for detecting and classifying an insider attack. A customized dataset from multiple files of the CERT dataset is used in this work. Four machine learning algorithms were applied to that dataset and gave better results. These algorithms are Random Forest, AdaBoost, XGBoost, and LightGBM. Using these supervised machine learning algorithms, this paper demonstrated the effective experimental results having higher accuracy in the classification report. Among the proposed algorithms, the LightGBM algorithm provides the highest accuracy of 97%; the other accuracy values are RF with 86%, AdaBoost with 88%, and XGBoost with 88.27%. In the future, the proposed models may increase their performance by expanding the dataset in size and diversity in terms of its features and the new trends of insider attackers to perform the attack. This may open up new research trends toward detecting and classifying insider attacks related to many fields of organization. Machine learning models are used by businesses to make credible business decisions, and improved model results lead to better judgments. The cost of mistakes can be quite high, however, this cost is reduced by improving model accuracy. ML-based research enables users to provide massive amounts of data to computer algorithms, which then evaluate, recommend, and decide using the supplied data.

REFERENCES

[1] U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, ‘‘Cloud-based email phishing attack using machine and deep learning algorithm,’’ Complex Intell. Syst., pp. 1–28, Jun. 2022.

[2] D. C. Le and A. N. Zincir-Heywood, ‘‘Machine learning based insider threat modelling and detection,’’ in Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM), Apr. 2019, pp. 1–6.

[3] P. Oberoi, ‘‘Survey of various security attacks in clouds based environments,’’ Int. J. Adv. Res. Comput. Sci., vol. 8, no. 9, pp. 405–410, Sep. 2017.

[4] A. Ajmal, S. Ibrar, and R. Amin, ‘‘Cloud computing platform: Performance analysis of prominent cryptographic algorithms,’’ Concurrency Comput., Pract. Exper., vol. 34, no. 15, p. e6938, Jul. 2022.

[5] U. A. Butt, R. Amin, M. Mehmood, H. Aldabbas, M. T. Alharbi, and N. Albaqami, ‘‘Cloud security threats and solutions: A survey,’’ Wireless Pers. Commun., vol. 128, no. 1, pp. 387–413, Jan. 2023.

[6] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman, and M. Bilal, ‘‘Smart home security: Challenges, issues and solutions at different IoT layers,’’ J. Supercomput., vol. 77, no. 12, pp. 14053–14089, Dec. 2021.

[7] S. Zou, H. Sun, G. Xu, and R. Quan, ‘‘Ensemble strategy for insider threat detection from user activity logs,’’ Comput., Mater. Continua, vol. 65, no. 2, pp. 1321–1334, 2020.

[8] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, ‘‘On the effectiveness of machine and deep learning for cyber security,’’ in Proc. 10th Int. Conf. Cyber Conflict (CyCon), May 2018, pp. 371–390.

[9] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, ‘‘Analyzing data granularity levels for insider threat detection using machine learning,’’ IEEE Trans. Netw. Service Manag., vol. 17, no. 1, pp. 30–44, Mar. 2020.