IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 1, August 2024

Predictive Shield: Harnessing Machine Learning to Forecast Vulnerability Exploitability

Dr Priya P Sajan[1], Sanketan Ashok Mohate[2], Sarthak Kishor Thorat[3], Shakeel Sheikh[4], Shivam Dilip Naik[5], Shivam Kailas Pagar[6]

Senior Project Engineer, C-DAC, Thiruvananthapuram, India[1]

PG-Diploma in Cyber Security and Forensics, C-DAC, Thiruvananthapuram, India[23456]

Abstract*: In today's world, cybersecurity risks are getting trickier. It's super important to think ahead about how vulnerable systems might be taken advantage of. This is all about making smart defense tactics. The goal here is to build a system that predicts how weak certain vulnerabilities can be when it comes to attacks. We're using the Common Vulnerability scoring System (CVSS) metrics for this task.*

By digging into a detailed dataset from the National Vulnerability Database (NVD), this project turns the data from JSON format into a CSV table. After that, it finds key characteristics and uses machine learning to guess how likely vulnerabilities are to be exploited. The process involves breaking down CVSS info to identify crucial parts like Attack Vector (AV), Attack Complexity (AC), Privileges Needed (PR), User Involvement (UI), Scope (S), Confidentiality (C), Integrity (I), and Availability (A). All these elements become inputs for our model, which we then tweak and check using various methods to ensure it's accurate & reliable.

The results reveal just how important the selected features & the predictive model are for calculating vulnerability susceptibility. This gives valuable insights for everyone in cybersecurity. Our initiative stresses the importance of preprocessing data, picking relevant features, and using predictive models to make cybersecurity strategies stronger. Going forward, we'll work on improving the model with more data & explore advanced algorithms to boost prediction accuracy. In short, our project shows how data-driven approaches can really help improve cybersecurity defenses and lessen the risks linked with exploitable weaknesses.

Keywords: Exploitability Prediction, Cybersecurity, Machine Learning, CVSS Metrics

I. INTRODUCTION

BACKGROUND AND CONTEXT OF THE PROJECT

In today's digital age, cybersecurity has emerged as a paramount concern for organizations and individuals alike. The rapid proliferation of digital technologies and the increasing reliance on interconnected systems have significantly expanded the attack surface, making cybersecurity more challenging than ever before. Cyber threats are not only growing in volume but also in sophistication, with adversaries constantly evolving their tactics to exploit vulnerabilities in software and systems. In this context, understanding and mitigating vulnerabilities has become a critical aspect of maintaining robust cybersecurity postures.

Vulnerabilities are weaknesses or flaws in software and systems that can be exploited by attackers to gain unauthorized access, disrupt operations, or steal sensitive information. The identification and remediation of these vulnerabilities are essential to prevent potential security breaches. Traditionally, vulnerability management has relied on reactive approaches, where vulnerabilities are patched after they are discovered and reported. However, this reactive strategy often leaves a window of opportunity for attackers to exploit these weaknesses before they are fixed.

To address this gap, the field of predictive analytics has gained traction in cybersecurity. Predictive analytics involves the use of statistical techniques, machine learning algorithms, and data mining to forecast future events based on historical data. In the realm of cybersecurity, predictive analytics can be leveraged to predict the exploitability of vulnerabilities, enabling organizations to proactively address the most critical threats before they are exploited. This

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, August 2024

project is situated within this innovative approach, aiming to develop a predictive model for vulnerability exploitability using data from the Common Vulnerability Scoring System (CVSS).

Importance and relevance of predicting exploitability

The ability to predict the exploitability of vulnerabilities holds immense importance and relevance in today's cybersecurity landscape. The sheer volume of vulnerabilities reported each year makes it impractical for organizations to address all of them immediately. According to the National Vulnerability Database (NVD), thousands of new vulnerabilities are disclosed annually, each varying in severity and potential impact. This deluge of vulnerabilities necessitates a prioritization strategy that focuses on the most critical and exploitable ones to optimize resource allocation and enhance security measures.

Predicting exploitability enables organizations to prioritize their remediation efforts effectively. Not all vulnerabilities pose the same level of risk; some are more likely to be exploited by attackers than others. By accurately predicting which vulnerabilities are most likely to be exploited, security teams can allocate their resources to address these high-risk vulnerabilities first, thereby reducing the overall attack surface and mitigating potential threats more efficiently. This proactive approach not only enhances the security posture but also minimizes the window of exposure, reducing the likelihood of successful attacks.

Moreover, predicting exploitability contributes to more informed decision-making in vulnerability management. Security teams often face the challenge of balancing the need to address vulnerabilities with the constraints of limited resources, such as time, budget, and personnel. A predictive model provides valuable insights into the exploitability likelihood, enabling organizations to make data-driven decisions about which vulnerabilities to prioritize for patching or mitigation. This strategic approach ensures that critical vulnerabilities are addressed promptly, reducing the risk of exploitation and potential damage.

Another significant aspect of predicting exploitability is its role in enhancing threat intelligence. Threat intelligence involves gathering, analyzing, and leveraging information about potential threats to inform security decisions. By integrating predictive models into threat intelligence workflows, organizations can gain a deeper understanding of the threat landscape and anticipate potential attacks. This foresight allows for the development of targeted defence strategies, improving the overall resilience of systems and networks against cyber threats.

Furthermore, predictive exploitability models can aid in compliance and regulatory adherence. Many industries are subject to stringent regulatory requirements that mandate the timely remediation of vulnerabilities to protect sensitive data and ensure the integrity of systems. By utilizing predictive analytics, organizations can demonstrate a proactive approach to vulnerability management, aligning with regulatory standards and reducing the risk of non-compliance penalties.

In addition to organizational benefits, predicting exploitability has broader implications for the cybersecurity community. The development and dissemination of predictive models can foster collaboration and knowledge sharing among cybersecurity professionals. As organizations share their predictive methodologies and insights, the collective understanding of vulnerability exploitability improves, leading to more effective and standardized approaches to vulnerability management across the industry.

Objective of the project

The primary objective of this project is to develop a robust and accurate predictive model for vulnerability exploitability using data from the CVSS. The CVSS is a widely adopted framework for assessing the severity of software vulnerabilities, providing a standardized way to evaluate their potential impact. By leveraging the rich dataset provided by the CVSS, this project aims to extract meaningful features and employ machine learning techniques to predict the likelihood of vulnerability exploitation.

To achieve this overarching objective, the project is structured around several specific goals:

- Data Collection and Preprocessing: The first goal is to collect a comprehensive dataset of vulnerabilities from the NVD, which includes detailed information about each vulnerability's CVSS metrics. This dataset will be preprocessed to ensure its quality and consistency, involving steps such as data cleaning, normalization, and feature extraction. The preprocessing phase is crucial for preparing the data for subsequent analysis and modeling.

- Feature Engineering: Feature engineering involves the selection and transformation of relevant features from the dataset that are likely to influence exploitability. This project will focus on extracting key attributes from the CVSS vectors, such as Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality (C), Integrity (I), and Availability (A). These features will serve as inputs to the predictive model.
- Model Development: The core of the project is the development of the predictive model. Various machine learning algorithms will be explored and evaluated to determine the most effective approach for predicting exploitability. This may include supervised learning techniques such as decision trees, random forests, support vector machines, and neural networks. The model will be trained and validated using appropriate evaluation metrics to ensure its accuracy and reliability.
- Model Evaluation and Validation: Once the model is developed, it will be rigorously evaluated and validated to assess its performance. This involves measuring metrics such as accuracy, precision, recall, and F1-score to determine how well the model predicts exploitability. Cross-validation techniques will be employed to ensure the model's generalizability and robustness.
- Result Analysis and Interpretation: The results obtained from the predictive model will be analyzed to gain insights into the factors that influence exploitability. This analysis will help in understanding the significance of different CVSS metrics and their contribution to the likelihood of exploitation. The findings will be interpreted in the context of existing cybersecurity knowledge and literature.
- Implementation and Deployment: The final goal is to implement and deploy the predictive model in a practical setting. This involves developing a user-friendly interface or integrating the model into existing security tools and workflows. The deployment phase aims to demonstrate the model's utility in real-world scenarios and its potential impact on enhancing vulnerability management practices.

By achieving these objectives, the project aims to contribute to the field of cybersecurity by providing a valuable tool for predicting vulnerability exploitability. The predictive model developed through this project will enable organizations to adopt a proactive approach to vulnerability management, prioritize critical threats, and enhance their overall security posture. Additionally, the project seeks to advance the understanding of the factors that influence exploitability, contributing to the broader body of knowledge in cybersecurity research. This project addresses the pressing need for predictive analytics in cybersecurity by focusing on the exploitability of vulnerabilities. By leveraging the CVSS dataset and employing machine learning techniques, the project aims to develop a robust predictive model that can inform vulnerability management strategies and improve the resilience of systems and networks against cyber threats. The importance and relevance of predicting exploitability cannot be overstated, as it enables organizations to proactively mitigate risks, optimize resource allocation, and enhance their overall cybersecurity posture. Through rigorous data collection, feature engineering, model development, evaluation, and deployment, this project aspires to make a meaningful contribution to the field of cybersecurity and support the ongoing efforts to safeguard digital assets in an increasingly interconnected world.

II. LITERATURE REVIEW

Overview of Existing Work Related to Exploitability Prediction

Cybersecurity has evolved significantly, driven by the increasing sophistication of cyber threats. As organizations become more dependent on digital systems, predicting the exploitability of vulnerabilities has become essential. The field of exploitability prediction, though relatively new, has gained traction due to its potential to enhance proactive security measures. This section reviews existing research, focusing on the models, algorithms, and approaches developed to predict and mitigate vulnerabilities.

The National Vulnerability Database (NVD) and the Common Vulnerability Scoring System (CVSS) have been instrumental in standardizing vulnerability assessment. CVSS metrics such as Attack Vector (AV), Attack Complexity (AC), and Confidentiality (C) are widely used in predictive models. Researchers have leveraged these metrics to build machine learning models that predict the likelihood of vulnerability exploitation.

Machine learning (ML) has been foundational in exploitability prediction. Supervised learning techniques, using historical data to train models, have been particularly successful. Decision trees and random forests are among the most popular algorithms due to their interpretability and effectiveness in handling complex datasets. Support vector machines (SVMs) are also commonly used, offering robust performance in scenarios with non-linear data relationships. More recently, neural networks, including deep learning models, have been explored for their ability to capture intricate patterns in large datasets.

Natural language processing (NLP) has further enhanced predictive models by extracting features from textual vulnerability descriptions. Techniques like tokenization and sentiment analysis are employed to convert unstructured text into structured data, which, when combined with CVSS metrics, improves prediction accuracy.

Unsupervised learning approaches, such as clustering algorithms like k-means and hierarchical clustering, have been used to identify patterns among vulnerabilities. These techniques group vulnerabilities with similar exploitability characteristics, aiding in targeted mitigation strategies.

The integration of threat intelligence data has added a dynamic element to exploitability prediction. By incorporating real-time information on active exploits and threat actor behavior, models can better contextualize vulnerabilities, leading to more timely and accurate predictions.

DATA FEED - NATIONAL VULNERABILITY DATABASE

The National Vulnerability Database (NVD) is a critical resource for cybersecurity professionals, providing comprehensive and standardized information on software vulnerabilities. The NVD offers various APIs and data feeds that facilitate access to its rich dataset, enabling users to stay up-to-date with the latest vulnerability information. The NVD feed table contains links and short descriptions for each API or data feed, guiding users on how to utilize these resources effectively. To leverage the NVD data, users must have an understanding of XML and/or JSON standards and related technologies as defined by the World Wide Web Consortium (www.w3.org).

Among the available data feeds, the CVE and CPE APIs are noteworthy for their flexibility and richness. These APIs offer an alternative to traditional vulnerability data feed files by providing a more versatile interface that includes a broader dataset. The APIs are updated as frequently as the NVD website, unlike the traditional feeds which have explicit update intervals. This frequent updating ensures that users have access to the most current information, which is crucial for timely vulnerability management.

The JSON Vulnerability Feeds provided by the NVD include detailed information for each vulnerability, such as descriptions, reference links from the CVE dictionary feed, CVSS base metrics, vulnerable product configurations, and weakness categorizations. These feeds enable users to perform advanced searches based on various criteria, such as CVE and CPE entries, and to view only the information that has changed since a given date or time. This capability simplifies the process of identifying relevant vulnerabilities and their associated risks.

For users who prefer to mirror NVD data locally, the traditional data feeds offer a structured way to stay synchronized. The vulnerability feeds provide CVE data organized by the first four digits of a CVE identifier, with separate feeds for each year. After an initial import of the complete dataset using the compressed JSON vulnerability feeds, users can utilize the “modified” feeds to keep their local database up-to-date. These feeds are updated nightly, but only if there are changes, ensuring efficient use of resources.

Each feed is accompanied by a META file, which users should consult to determine if updates have occurred since the last import. This approach minimizes unnecessary downloads and optimizes the synchronization process. By leveraging these APIs and data feeds, cybersecurity professionals can maintain an up-to-date and comprehensive understanding of vulnerabilities, enhancing their ability to predict and mitigate exploitability.

Key Models, Algorithms, and Approaches Previously Used

- Decision Trees and Random Forests: These models are favored for their simplicity and ability to handle complex datasets. Decision trees split data based on feature values, while random forests aggregate multiple decision trees to reduce overfitting and improve generalization.

- Support Vector Machines (SVMs): SVMs are effective for classification tasks in exploitability prediction. They work by finding the optimal hyperplane that separates data into classes, making them suitable for datasets with non-linear relationships.
- Neural Networks and Deep Learning: These models are increasingly popular for their ability to analyze large and complex datasets. Convolutional Neural Networks (CNNs) are used for structured data, while Recurrent Neural Networks (RNNs) are applied to sequential data, offering a powerful tool for predicting exploitability.
- Natural Language Processing (NLP) Techniques: NLP has been crucial in processing textual descriptions of vulnerabilities. Advanced models like transformers and BERT have significantly improved the extraction of contextual features, enhancing exploitability prediction.
- Clustering Algorithms: Techniques such as k-means and hierarchical clustering help identify common characteristics among vulnerabilities, providing insights into high-risk clusters.
- Threat Intelligence Integration: Incorporating threat intelligence feeds into predictive models adds real-time context, improving the accuracy and timeliness of exploitability predictions.
- Hybrid Approaches: Combining multiple models and techniques, such as ML and NLP, has proven effective in enhancing exploitability prediction. These hybrid models leverage both structured and unstructured data for more comprehensive predictions.
- Feature Selection and Engineering: The selection and engineering of features are crucial for model performance. Techniques like Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) are used to refine features, improving model accuracy.

III. RESEARCH METHODOLOGY

DESCRIPTION OF THE DATASET USED

The dataset used in this project is derived from the National Vulnerability Database (NVD), which is a comprehensive repository of standardized information on software vulnerabilities. The NVD includes detailed data on each vulnerability, including descriptive information, associated reference links, and metrics based on the Common Vulnerability Scoring System (CVSS). The CVSS provides a consistent method for assessing the severity of vulnerabilities through various metrics that capture both the exploitability and impact of each vulnerability. For this project, the dataset encompasses multiple features extracted from the CVSS metrics, focusing on elements that are crucial for predicting the exploitability of vulnerabilities.

The source of the data is the NVD's JSON vulnerability feeds, which offer a rich and flexible dataset. These feeds include a wide array of details for each vulnerability, such as the CVE identifier, vulnerability descriptions, CVSS base metrics, configurations of vulnerable products, and categorizations of weaknesses. The JSON format of the data allows for comprehensive coverage and ease of parsing, making it suitable for our analytical purposes. Given the dynamic nature of vulnerabilities, the dataset is periodically updated to reflect the latest information, ensuring that the model can be trained on current and relevant data.

Preprocessing steps are essential to prepare the raw data for analysis and model training. The preprocessing phase begins with data extraction, where the JSON files from the NVD are parsed to extract relevant fields. This is followed by data cleaning, which involves handling missing values, correcting inconsistencies, and removing any duplicate entries. Given that the CVSS metrics are central to our predictive model, these metrics are specifically targeted during preprocessing to ensure their accuracy and completeness. The key metrics extracted include Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality (C), Integrity (I), and Availability (A). Each of these metrics is converted into a suitable numerical format to facilitate machine learning processes.

Normalization is another critical preprocessing step, particularly given the diverse scales of the CVSS metrics. Normalization ensures that all features contribute equally to the model, preventing any single metric from disproportionately influencing the predictions. Techniques such as min-max scaling or z-score normalization can be employed to standardize the features. Additionally, categorical features such as the attack vector and user interaction, which are represented by distinct categories, are encoded using techniques like one-hot encoding. This process transforms categorical data into a numerical format, making it compatible with machine learning algorithms.

EXPLANATION OF THE MODEL SELECTION

The core of our predictive modeling effort is the RandomForestRegressor algorithm, which is a popular and powerful ensemble learning technique. The decision to use the RandomForestRegressor is based on its ability to handle high-dimensional data, manage interactions between features effectively, and provide robust predictions even when faced with complex and noisy datasets. Random forests are particularly well-suited for this project because they mitigate the risk of overfitting, a common issue in machine learning, by aggregating the predictions of multiple decision trees.

The RandomForestRegressor algorithm operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees for regression tasks. Each decision tree is trained on a bootstrap sample of the data, with a random subset of features considered at each split, ensuring diversity among the trees. This ensemble approach leverages the strengths of individual trees while compensating for their weaknesses, resulting in improved generalization and predictive performance.

The choice of RandomForestRegressor also stems from its interpretability and ease of implementation. Despite being an ensemble method, random forests offer insights into feature importance, allowing us to understand which features contribute most to the prediction of exploitability. This transparency is valuable in the context of cybersecurity, where understanding the factors influencing predictions can inform better decision-making and risk management.

DESCRIPTION OF THE IMPLEMENTATION STEPS

The implementation of the predictive model involves several key steps: feature selection, model training, and evaluation metrics. Each step is meticulously designed to ensure that the model is both accurate and reliable in predicting the exploitability of vulnerabilities.

Feature Selection

Feature selection is a crucial step in the modeling process as it involves identifying the most relevant features that contribute to predicting exploitability. Given the comprehensive nature of the CVSS metrics, the primary features considered include Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality (C), Integrity (I), and Availability (A). These features are selected based on their theoretical importance and empirical evidence from previous research, which indicates their strong influence on exploitability.

To further refine the feature set, techniques such as recursive feature elimination (RFE) and feature importance analysis from the RandomForestRegressor can be employed. RFE iteratively removes the least important features based on model performance until the optimal subset of features is identified. Feature importance analysis leverages the inherent capabilities of random forests to rank features based on their contribution to the predictive power of the model. These methods ensure that only the most impactful features are retained, enhancing model efficiency and accuracy.

Model Training

Once the features are selected, the next step is to train the RandomForestRegressor model. The training process involves splitting the dataset into training and validation sets to evaluate the model's performance on unseen data. Typically, a common split ratio is 80-20, where 80% of the data is used for training and 20% for validation. This split ensures that the model can learn from a substantial portion of the data while being tested on a meaningful sample of unseen instances.

The training process involves tuning various hyperparameters of the RandomForestRegressor to optimize its performance. Key hyperparameters include the number of trees in the forest (n_estimators), the maximum depth of each tree (max_depth), the minimum number of samples required to split an internal node (min_samples_split), and the minimum number of samples required to be at a leaf node (min_samples_leaf). Hyperparameter tuning can be performed using techniques such as grid search or random search, which systematically explore different combinations of hyperparameters to identify the best configuration.

During training, the RandomForestRegressor constructs multiple decision trees, each trained on a bootstrap sample of the data. At each node of the tree, a random subset of features is considered for splitting, ensuring diversity among the trees. The aggregation of predictions from all the trees in the forest leads to the final prediction, leveraging the ensemble approach to enhance accuracy and robustness.

Evaluation Metrics

Evaluating the performance of the predictive model is essential to ensure its reliability and effectiveness. Several metrics are used to assess the model's performance, focusing on both accuracy and robustness. Key evaluation metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) score.

Mean Squared Error (MSE): MSE measures the average squared difference between the actual and predicted values. It penalizes larger errors more severely, making it sensitive to outliers. A lower MSE indicates better model performance.

Root Mean Squared Error (RMSE): RMSE is the square root of MSE, providing a measure of the average magnitude of the prediction errors. Like MSE, a lower RMSE signifies better predictive accuracy.

Mean Absolute Error (MAE): MAE calculates the average absolute difference between the actual and predicted values, providing a straightforward measure of prediction accuracy. Unlike MSE and RMSE, MAE treats all errors equally, making it less sensitive to outliers.

R-squared (R^2) score: The R^2 score represents the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 score close to 1 indicates that the model explains a large portion of the variance, reflecting high predictive power.

The evaluation process involves computing these metrics on the validation set to assess the model's performance on unseen data. Additionally, cross-validation techniques can be employed to further validate the model's robustness. Cross-validation involves partitioning the dataset into multiple folds, training the model on each fold, and evaluating it on the remaining data. This process provides a more comprehensive assessment of the model's performance and helps to mitigate the risk of overfitting.

The methodology for this project encompasses a systematic approach to data preparation, model selection, and implementation, ensuring that the predictive model is both accurate and reliable. The use of the RandomForestRegressor algorithm, coupled with careful feature selection and rigorous evaluation metrics, provides a robust framework for predicting the exploitability of vulnerabilities. By leveraging the rich dataset from the NVD and employing advanced machine learning techniques, this project aims to enhance the effectiveness of vulnerability management and contribute to the broader field of cybersecurity.

IV. IMPLEMENTAION

EXPLANATION OF THE CODE AND FUNCTIONS USED

The implementation of the predictive model for vulnerability exploitability involves several critical stages, each of which is supported by specific functions and code segments. This section provides a comprehensive explanation of the code and functions utilized in the project, highlighting the key segments and discussing the challenges encountered during implementation.

The primary programming language used for this project is Python, leveraging its extensive libraries and frameworks for data analysis and machine learning, such as pandas, scikit-learn, and JSON handling utilities. The process begins with data extraction from the National Vulnerability Database (NVD), followed by preprocessing, feature selection, model training, and evaluation.

1. Importing Necessary Libraries

The first step involves importing essential Python libraries needed for data processing:

json: For handling JSON data files.

csv: For writing data to a CSV file.

re: For using regular expressions to parse the CVSS vector.

‘‘‘

```
import json
import csv
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, August 2024

```
import re
‘‘‘
```

2. Defining the Function to Parse the CVSS Vector

We create a function named parse_cvss_vector to process and extract specific components from the CVSS vector string. The function uses regular expressions to identify key-value pairs (e.g., AV:N where AV is the key and N is the value).

```
‘‘‘
def parse_cvss_vector(cvss_vector):
  pattern = re.compile(r'([A-Z]+):([A-Z]+)')
  return dict(pattern.findall(cvss_vector))
‘‘‘
```

3. Converting JSON Data to CSV

The convert_nvd_json_to_csv function is defined to convert the JSON data (from the National Vulnerability Database) into a CSV format. This function reads the JSON file, extracts relevant data, and writes it into a CSV file.

```
‘‘‘
def convert_nvd_json_to_csv(json_file, csv_file):
  with open(json_file, 'r') as f:
    data = json.load(f)
‘‘‘
```

a) Defining CSV Columns

The CSV file will contain specific columns, such as the CVE ID, description, CVSS score, and various metrics (AV, AC, PR, UI, S, C, I, A). These columns are defined as a list of strings.

```
‘‘‘
csv_columns = [
    'cve_id', 'description', 'cvss_score',
    'AV', 'AC', 'PR', 'UI', 'S', 'C', 'I', 'A'
  ]
‘‘‘
```

b) Writing Data to CSV

The function opens the CSV file for writing and uses csv.DictWriter to handle writing rows to the file. A header row is written first.

```
‘‘‘
  with open(csv_file, 'w', newline='') as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=csv_columns)
writer.writeheader()
‘‘‘
```

c) Processing Each CVE Entry

The function then loops over each item in the JSON data (i.e., each CVE entry). For each entry, it extracts the CVE ID,

description, CVSS score, and CVSS vector string. The parse_cvss_vector function is used to parse the vector string into individual metrics.

```
‘‘‘
    for item in data['CVE_Items']:
cve_id = item['cve']['CVE_data_meta']['ID']
```

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 1, August 2024

```
        description = item['cve']['description']['description_data'][0]['value']
```

d) Extracting CVSS Details

The script checks whether the CVSS version 3 metrics are available; if not, it defaults to version 2 metrics. The corresponding CVSS vector string is then parsed to extract the various metrics.

```
cvss_score = None
cvss_vector = None
        metrics = {}

        impact = item.get('impact')
        if impact:
            baseMetricV3 = impact.get('baseMetricV3')
            if baseMetricV3:
cvss_score = baseMetricV3['cvssV3']['baseScore']
cvss_vector = baseMetricV3['cvssV3']['vectorString']
                metrics = parse_cvss_vector(cvss_vector)
            else:
                baseMetricV2 = impact.get('baseMetricV2')
                if baseMetricV2:
cvss_score = baseMetricV2['cvssV2']['baseScore']
cvss_vector = baseMetricV2['cvssV2']['vectorString']
                    metrics = parse_cvss_vector(cvss_vector)
```

e) Creating a Row for the CSV

A dictionary is created to represent each row in the CSV file, containing all the extracted fields and their respective values. This row is then written to the CSV file.

```
        row = {
            'cve_id': cve_id,
            'description': description,
            'cvss_score': cvss_score,
            'AV': metrics.get('AV', ''),
            'AC': metrics.get('AC', ''),
            'PR': metrics.get('PR', ''),
            'UI': metrics.get('UI', ''),
            'S': metrics.get('S', ''),
            'C': metrics.get('C', ''),
            'I': metrics.get('I', ''),
            'A': metrics.get('A', '')
        }

writer.writerow(row)
```

4. Executing the Conversion Process

Finally, the script calls the convert_nvd_json_to_csv function, passing the path to the JSON file and the desired output path for the CSV file. This step triggers the conversion process.

‘‘‘

```
json_file = '/content/nvdcve-1.1-modified.json'
csv_file = '/content/cve_dataset.csv'
convert_nvd_json_to_csv(json_file, csv_file)
```

1. Importing Required Libraries

The implementation begins with importing essential libraries for data processing, model training, evaluation, saving the model, and handling web requests.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import joblib
import requests
```

2. Step 1: Collecting the Dataset

The dataset is loaded from a CSV file using the pandas library. This dataset contains various attributes of CVEs and is crucial for building the prediction model.

```
dataset_path = '/content/cve_dataset.csv'
df = pd.read_csv(dataset_path)
```

3. Step 2: Data Preprocessing

Before training the model, data preprocessing is performed:

a) Handling Missing Data

Rows with missing or empty fields in the vector string fields (AV, AC, PR, UI, S, C, I, A) are removed to ensure data quality.

```
df.dropna(subset=['AV', 'AC', 'PR', 'UI', 'S', 'C', 'I', 'A'], inplace=True)
df = df[(df['AV'] != '') & (df['AC'] != '') & (df['PR'] != '') & (df['UI'] != '') &
        (df['S'] != '') & (df['C'] != '') & (df['I'] != '') & (df['A'] != '')]
```

b) Converting Categorical Variables

Categorical variables in the dataset are converted into dummy/indicator variables using one-hot encoding. This step transforms the categorical values into a format that can be used in the machine learning model

```
df = pd.get_dummies(df, columns=['AV', 'AC', 'PR', 'UI', 'S', 'C', 'I', 'A'])
```

c) Separating Features and Labels

The dataset is then split into features (X) and labels (y). Features include all the predictor variables, while the label is the CVSS score we aim to predict.

```
X = df.drop(columns=['cve_id', 'description', 'cvss_score'])
y = df['cvss_score']
```

4. Step 3: Training a Machine Learning Model

a) Splitting the Dataset

The dataset is divided into training and testing sets, where 80% of the data is used for training and 20% for testing. This split helps in evaluating the model's performance on unseen data.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, August 2024

‘‘‘
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```
‘‘‘

b) Training the RandomForestRegressor Model
A RandomForestRegressor is instantiated and trained on the training data. This model is chosen for its robustness and ability to handle large datasets.

‘‘‘
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```
‘‘‘

c) Evaluating the Model
After training, the model's performance is evaluated using the test set. Two metrics are calculated: Mean Squared Error (MSE) and R-squared (R^2). MSE measures the average squared difference between actual and predicted values, while R^2 indicates the proportion of variance explained by the model.

‘‘‘
```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Model Mean Squared Error: {mse:.2f}")
print(f"Model R-squared: {r2:.2f}")
```
‘‘‘

d) Saving the Trained Model
The trained model is saved to a file using joblib, allowing it to be loaded later for predictions without retraining.

‘‘‘
```
joblib.dump(model, 'exploitability_model.pkl')
```
‘‘‘

5. Step 4: Predicting CVSS Scores
a) Function to Retrieve CVE Data
The get_cvss_score function retrieves the CVE data for a specific CVE ID from the NVD (National Vulnerability Database) API. It extracts the CVSS vector string, which is used to generate the features for prediction.

‘‘‘
```
def get_cvss_score(cve_id):
  try:
url = f'https://services.nvd.nist.gov/rest/json/cves/2.0?cveId={cve_id}'
    response = requests.get(url)
response.raise_for_status()

cve_data = response.json()

    if cve_data and 'vulnerabilities' in cve_data:
cve_item = cve_data['vulnerabilities'][0]['cve']
      metrics = cve_item.get('metrics', {}).get('cvssMetricV31', [])

      if not metrics:
        return None

      cvss_v3 = metrics[0].get('cvssData', {})
vector_string = cvss_v3.get('vectorString', '')
```

```
            # Parse the vector string
vector_components = vector_string.split('/')
vector_dict = {}
            for component in vector_components[1:]:
                key, value = component.split(':')
vector_dict[key] = value

            return {
                'AV': vector_dict.get('AV', ''),
                'AC': vector_dict.get('AC', ''),
                'PR': vector_dict.get('PR', ''),
                'UI': vector_dict.get('UI', ''),
                'S': vector_dict.get('S', ''),
                'C': vector_dict.get('C', ''),
                'I': vector_dict.get('I', ''),
                'A': vector_dict.get('A', ''),
            }
        else:
            return None
    except requests.exceptions.RequestException as e:
print(f"An error occurred while retrieving CVE data: {e}")
        return None
```

b) Function to Predict CVSS Score

The predict_cvss_score function loads the trained model and predicts the CVSS score for a given CVE ID using the feature vector extracted from the CVE data.

```
def predict_cvss_score(cve_id):
cve_data = get_cvss_score(cve_id)

    if cve_data is None or any(value == '' for value in cve_data.values()):
        return None

    # Load the trained model
    model = joblib.load('exploitability_model.pkl')

    # Prepare the feature vector for prediction
feature_vector = pd.DataFrame([cve_data])
feature_vector = pd.get_dummies(feature_vector, columns=['AV', 'AC', 'PR', 'UI', 'S', 'C', 'I', 'A'])

    # Ensure the feature vector has the same columns as the training data
    for column in X.columns:
        if column not in feature_vector.columns:
feature_vector[column] = 0

    # Reorder columns to match the training data
feature_vector = feature_vector[X.columns]

    # Predict CVSS score
    prediction = model.predict(feature_vector)

    return prediction[0]
```

c) Suggesting Remediation Actions

Based on the predicted CVSS score, the suggest_remediation_action function suggests appropriate remediation actions and classifies the severity of the vulnerability.

```
def suggest_remediation_action(cvss_score):
    if cvss_score>= 9.0:
        action = "Immediate attention and remediation required. Apply patches or mitigation measures urgently."
        severity = "Critical"
elifcvss_score>= 7.0:
        action = "High priority for remediation. Plan and apply patches or mitigation measures as soon as possible."
        severity = "High"
elifcvss_score>= 4.0:
        action = "Important to address, but may not require immediate action. Schedule remediation within a reasonable
time frame."
        severity = "Medium"
elifcvss_score>= 0.1:
        action = "Monitor and address as part of regular maintenance. Remediate based on business priorities and available
resources."
        severity = "Low"
    else:
        action = "No action required."
        severity = "None"

    return severity, action
```

d) Example Use Case

Finally, the code demonstrates an example use case where a specific CVE ID is provided, and the model predicts its CVSS score along with suggested remediation actions.

```
cve_id = 'CVE-2024-5855'  # Note: Replace with the CVE ID you want to predict CVSS score for
cvss_score_prediction = predict_cvss_score(cve_id)

if cvss_score_prediction is not None:
print(f"The predicted CVSS score for CVE {cve_id} is {cvss_score_prediction:.2f}.")
else:
print(f"CVSS score prediction for {cve_id} is not available.")

cvss_score = cvss_score_prediction
severity, action = suggest_remediation_action(cvss_score)
print(f"Severity: {severity}")
print(f"Suggested Remediation Action: {action}")

```

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 1, August 2024

V. RESULT

The predictive model for vulnerability exploitability, based on the RandomForestRegressor algorithm, demonstrates exceptional performance and reliability. The model evaluation metrics provide a clear indication of its accuracy and effectiveness. The Mean Squared Error (MSE) is a low 0.08, indicating that the average squared difference between the predicted and actual CVSS scores is minimal. This low MSE value reflects the model's precision in predicting exploitability scores. The R-squared (R^2) value is an impressive 0.97, signifying that the model explains 97% of the variance in the CVSS scores, thereby highlighting its strong predictive capability.

To illustrate the model's practical application, consider the example of CVE-2024-5855. The model predicts a CVSS score of 4.39 for this particular vulnerability. According to the CVSS classification, this score corresponds to a "Medium" severity level. Consequently, the suggested remediation action for this vulnerability is to address it with importance but not necessarily immediate urgency. The recommendation is to schedule remediation within a reasonable timeframe, ensuring that the vulnerability is mitigated before it can be exploited.

The results of the predictive model underscore its high accuracy and robustness in predicting the exploitability of vulnerabilities. The low MSE of 0.08 signifies that the model's predictions are closely aligned with the actual CVSS scores, which is crucial for making reliable assessments of vulnerability severity. The R^2 value of 0.97 further emphasizes the model's effectiveness in capturing the variability in the data, indicating that it accounts for nearly all the factors influencing the CVSS scores.

The practical implications of these results are significant for cybersecurity management. The model's ability to accurately predict CVSS scores enables organizations to prioritize their remediation efforts based on the predicted exploitability of vulnerabilities. This prioritization is vital because it allows security teams to allocate their limited resources more efficiently, focusing on vulnerabilities that pose the highest risk. By addressing the most critical vulnerabilities first, organizations can significantly reduce their exposure to potential attacks and enhance their overall security posture.

The predicted score for CVE-2024-5855 serves as a concrete example of the model's utility. With a predicted CVSS score of 4.39, the model classifies the vulnerability as having medium severity. This classification guides the remediation strategy, suggesting that while the vulnerability is important to address, it may not require immediate action. Instead, remediation can be scheduled within a reasonable timeframe, balancing the need to address the vulnerability with other operational priorities.

The results also highlight the importance of specific CVSS metrics in predicting exploitability. Metrics such as Attack Vector (AV), Attack Complexity (AC), and Privileges Required (PR) are particularly influential. For instance, vulnerabilities with a network attack vector are generally more exploitable than those requiring physical access. Similarly, vulnerabilities with low attack complexity and no required privileges are more likely to be exploited. Understanding the significance of these metrics helps in interpreting the model's predictions and making informed decisions about vulnerability management.

The results of the predictive model for vulnerability exploitability demonstrate its high accuracy and practical value. The low MSE and high R^2 values reflect the model's precision and robustness, making it a reliable tool for assessing vulnerability severity. The practical example of CVE-2024-5855 illustrates how the model's predictions can guide remediation efforts, helping organizations prioritize their security actions effectively. By leveraging the model's insights, organizations can enhance their cybersecurity posture, reduce the risk of exploitation, and allocate their resources more efficiently.

VI. CONCLUSION

The predictive model for vulnerability exploitability, developed using the RandomForestRegressor algorithm, has demonstrated exceptional accuracy and reliability. Key findings include a low Mean Squared Error (MSE) of 0.08 and a high R-squared (R^2) value of 0.97, indicating the model's capability to predict CVSS scores with remarkable precision. The model's ability to accurately predict scores, such as the 4.39 score for CVE-2024-5855, and its classification within the "Medium" severity range, underscores its practical utility in guiding cybersecurity efforts.

The significance of these results lies in their potential to transform vulnerability management practices. By providing precise predictions of exploitability, the model enables organizations to prioritize remediation efforts more effectively, thus enhancing their overall security posture. This prioritization is crucial in efficiently allocating limited resources to address the most critical vulnerabilities first, thereby reducing the risk of successful attacks.

Future work should focus on addressing the limitations identified in this study. This includes incorporating additional data sources, such as real-time threat intelligence feeds, to further enhance the model's predictive capabilities. Continuous validation and retraining of the model with updated data are also essential to maintain its accuracy and relevance in the evolving threat landscape. Moreover, exploring other machine learning algorithms and hybrid approaches could provide additional insights and improvements. By refining and expanding upon this work, the field of vulnerability exploitability prediction can continue to advance, offering even more robust tools for cybersecurity professionals.

REFERENCES

[1]. Alamro, H., Mtouaa, W., Aljameel, S., Salama, A. S., Hamza, M. A., & Othman, A. Y. (2023). Automated android malware detection using optimal ensemble learning approach for cybersecurity. IEEE Access.

[2]. Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A. V., & Di Franco, F. (2023). The role of machine learning in cybersecurity. Digital Threats: Research and Practice.

[3]. Breiman, L. (2001). Random forests. Machine learning.

[4]. Chen, T., &Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acmsigkdd international conference on knowledge discovery and data mining .

[5]. Dasgupta, D., Akhtar, Z., & Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. The Journal of DefenseModeling and Simulation.

[6]. Dutta, V., Choraś, M., Pawlicki, M., & Kozik, R. (2020). A deep learning ensemble for network anomaly and cyber-attack detection. Sensors.

[7]. Kuehn, P., Relke, D. N., & Reuter, C. (2023). Common vulnerability scoring system prediction based on open source intelligence information sources. Computers & Security.

[8]. Lin, G., Xiao, W., Zhang, L. Y., Gao, S., Tai, Y., & Zhang, J. (2021). Deep neural-based vulnerability discovery demystified: data, model and performance. Neural Computing and Applications.

[9]. Mell, P., Scarfone, K., & Romanosky, S. (2007, June). A complete guide to the common vulnerability scoring system version 2.0. In Published by FIRST-forum of incident response and security teams.

[10]. National Institute of Standards and Technology. (2024). National Vulnerability Database (NVD). Retrieved from https://nvd.nist.gov

[11]. Okoli, U. I., Obi, O. C., Adewusi, A. O., & Abrahams, T. O. (2024). Machine learning in cybersecurity: A review of threat detection and defense mechanisms. World Journal of Advanced Research and Reviews.

[12]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... &Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research.

[13]. Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K., & Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Engineering Journal