# Predictive Vulnerability Management: Using Machine Learning to Predict and Prioritize Vulnerability Exploits

Sruthika Sivakumar*, Vedansh Maheshwari†, Vanshaj‡
*USN: 1RVU22CSE164, School of Computer Science and Engineering, RV University, Bangalore, India
†USN: 1RVU22CSE187, School of Computer Science and Engineering, RV University, Bangalore, India
‡USN: 1RVU23CSE526, School of Computer Science and Engineering, RV University, Bangalore, India

## I. Introduction

In the rapidly evolving cybersecurity landscape, organizations face an increasing number of threats exploiting system vulnerabilities. The Common Vulnerabilities and Exposures (CVE) database, maintained by the National Vulnerability Database (NVD), is a critical resource for identifying these vulnerabilities. However, the vast amount of data poses challenges for manual assessment and prioritization. This project leverages machine learning to predict and prioritize vulnerability exploits, transitioning from reactive to proactive security measures. This enhances an organization's ability to efficiently allocate resources and mitigate potential damages.

The primary objective is to develop a machine learning model that accurately predicts and prioritizes vulnerability exploits using CVE data. Key tasks include:

1) Gathering and preprocessing CVE data.
2) Conducting exploratory data analysis (EDA).
3) Selecting and implementing appropriate machine learning models.
4) Training and evaluating these models.
5) Providing actionable insights for vulnerability management.

Achieving these objectives will result in a robust tool to assist cybersecurity professionals in making informed decisions, thereby strengthening overall security measures.

This report covers:

- **Data Identification and Gathering:** Data sources and collection methods.
- **Exploratory Data Analysis:** Statistical analysis and visualizations of data.
- **Data Preprocessing:** Techniques for data cleaning, normalization, and transformation.
- **Model Selection:** Evaluation of machine learning algorithms and justification for chosen models.
- **Training and Evaluation:** Methods for model training, hyperparameter tuning, and performance evaluation.
- **Results and Discussion:** Findings, model performance comparison, and insights.
- **Conclusion:** Project outcomes, limitations, and future work suggestions.

## II. Data Collection and Cleaning

For this project, data was sourced from the Vulners API[1] and the National Vulnerability Database (NVD).[2] The Vulners API provides detailed vulnerability data, including descriptions, CWE codes, and metadata. The NVD offers standardized and reliable CVE data. Using both sources ensured a comprehensive and accurate dataset for our machine learning models.

### A. Data Collection Process

- **Vulners API**: Retrieved detailed vulnerability data, including CWE codes and reference links.
- **National Vulnerability Database (NVD)**: Obtained standardized CVE data.

### B. Data Cleaning and Preprocessing

Key preprocessing steps included:

1) **Handling Missing Values**: Dropped rows with missing values to maintain data integrity.
2) **Renaming Columns**: Renamed `cve.CVE_data` to `cve_id` and reordered columns.
3) **Date Conversion**: Converted `publishedDate` and `lastModifiedDate` to datetime format.
4) **Extracting Nested Data**: Extracted CWE codes and descriptions from nested JSON structures.
5) **ExploitDB Reference Flag**: Added a flag to indicate references to ExploitDB.
6) **Counting References**: Counted the number of references for each CVE.

### C. Summary of Preprocessing Steps

- **Data Integrity**: Removed rows with missing values.
- **Column Standardization**: Renamed and reordered columns.
- **Date Handling**: Converted date columns to datetime format.

- **Nested Data Extraction**: Extracted CWE codes and descriptions.
- **Exploit Indicators**: Added flags and counts for ExploitDB references.

These steps ensured a well-structured and comprehensive dataset, ready for exploratory data analysis and model development.

## III. Exploratory Data Analysis

### Univariate Analysis

#### A. Distribution Plots

The distribution plots in Figure 1 provide insights into various attributes of the dataset. Key observations include:

- **CVSS Base Score**: Scores are evenly spread with peaks around 0.3 and 0.8.
- **Exploitability Score**: Peaks are noticeable at 0.2 and 0.9.
- **Impact Score**: Exhibits multiple peaks indicating varied impact severity.
- **Number of References**: Most CVEs have low references with some outliers.
- **ExploitDB Flag**: Few CVEs have associated exploits.
- **Severity Levels**: Medium severity is most common, followed by high and low.
- **AC Insufficient Information**: Most CVEs lack insufficient access control info.
- **Privileges Required**: Most CVEs do not require specific privileges.
- **User Interaction Required**: Majority do not require user interaction.
- **Publication Date**: Concentration around 2019 and 2020.
- **Publication Month**: Fairly even distribution with slight variations.

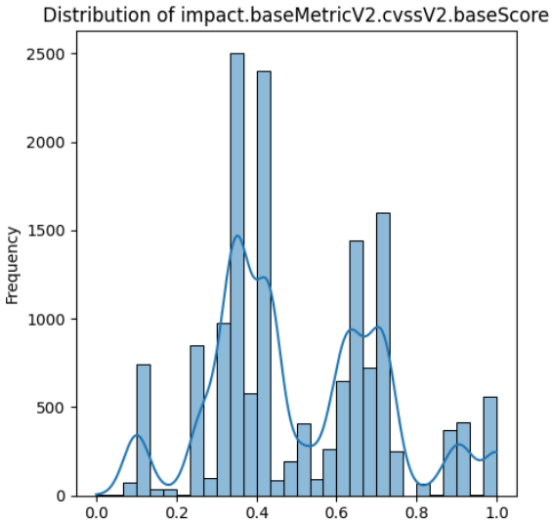


Fig. 1. Distribution Graph

#### B. Box Plots

The box plots in Figure 2 summarize the distributions and highlight outliers:

- **Scores**: CVSS base, exploitability, and impact scores show central tendencies with outliers, especially in exploitability.
- **Number of References**: Most CVEs have low references, with significant outliers.
- **ExploitDB Flag and Severity Levels**: Confirm rarity of ExploitDB entries and prevalence of medium severity.
- **AC Insufficient Information, Privileges Required, User Interaction**: Most CVEs do not require these conditions, with few outliers.
- **Publication Date and Month**: Show spread of publication times without significant outliers in months.

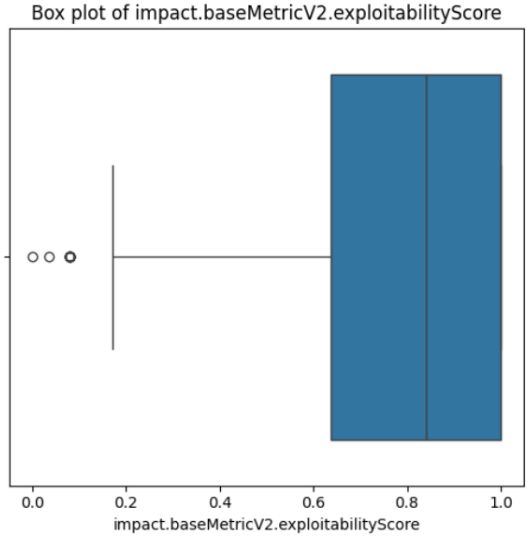


Fig. 2. Box-Plot

### Bivariate Analysis

#### C. Correlation Heatmap

The correlation heatmap in Figure 3 provides insights into the relationships between various features in our dataset:

- **Base Score and Exploitability Score**: High positive correlation (0.86) indicates that vulnerabilities with higher base scores are likely to have higher exploitability scores.
- **Base Score and Impact Score**: Strong negative correlation (-0.86) suggests that as the base score increases, the impact score tends to decrease.
- **Severity Levels**:
  - `severity_LOW` is negatively correlated with both `baseScore` and `exploitabilityScore`.
  - `severity_MEDIUM` shows a positive correlation with `exploitabilityScore`.
- **ExploitDB Flag**: Low correlation with most features, indicating its relative independence.

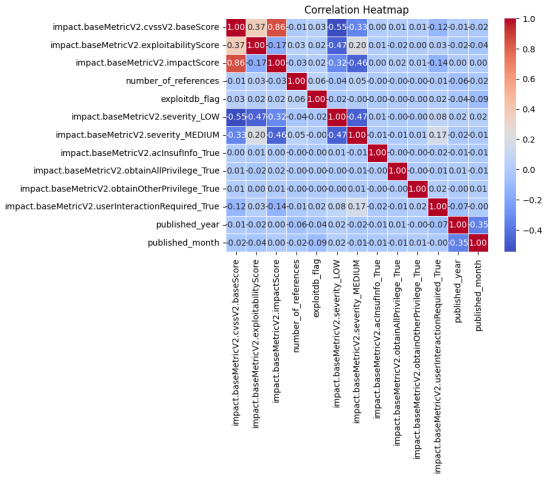


Fig. 3. Correlation Heatmap

### D. Base Score vs Exploitability Score Scatter Plot

The scatter plot between base score and exploitability score in Figure 4highlights:

- **Positive Relationship**: Higher base scores generally correspond to higher exploitability scores, reinforcing the high correlation seen in the heatmap.
- **Cluster Formation**: Data points form clusters at certain score combinations, indicating common scoring patterns.
- **Score Range**: Base scores range from approximately 2 to 10, while exploitability scores range from 1 to 10, showing variability in both severity and exploitability.

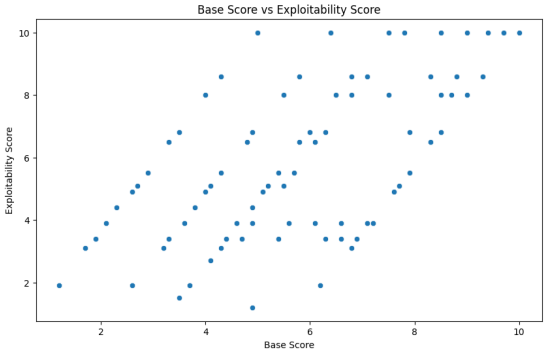


Fig. 4. Scatter Plot

This analysis helps in understanding the direct relationship between these two critical metrics, aiding in the development of predictive models for vulnerability risk assessment.

## Advanced Analysis and Visualizations

### E. Number of CVEs per Severity Level

The bar graph in Figure 5 illustrates the distribution of Common Vulnerabilities and Exposures (CVEs) across different severity levels: High, Medium, and Low.

- **High Severity**: This category includes vulnerabilities that pose significant risks and can potentially cause substantial harm if exploited. The graph shows that there are around 4,000 CVEs classified as high severity.
- **Medium Severity**: Vulnerabilities in this category can cause moderate damage and are the most prevalent in the dataset. The graph indicates approximately 9,000 CVEs of medium severity, making it the most common severity level.
- **Low Severity**: These vulnerabilities pose minimal risk and are less likely to be exploited. The graph displays around 2,000 CVEs classified as low severity.

This distribution highlights that medium severity vulnerabilities are the most common, followed by high and low severity vulnerabilities. Understanding the distribution of CVEs by severity helps prioritize which vulnerabilities to address first, focusing on those with the highest potential impact.

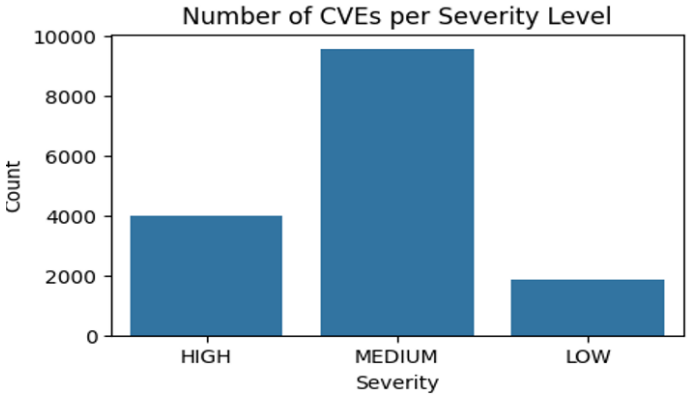


Fig. 5. Number of CVEs per Severity Level

### F. Violin Plot: Base Scores across Severity Levels

The violin plot depicts the distribution of base scores across different severity levels:

- **High Severity**: Base scores predominantly cluster around 9-10, indicating higher scores for high severity vulnerabilities.
- **Medium Severity**: Base scores are varied, clustering around 4-6.
- **Low Severity**: Base scores generally cluster around 2-3.

This visualization shows the significant variance in base scores across severity levels, highlighting the risk levels associated with different vulnerabilities.

### G. Grouping by Exploitability Score

The dataset was analyzed by grouping vulnerabilities based on their exploitability score and calculating the mean for numerical columns:

- **Grouping Method**: Used `groupby` on the `impact.baseMetricV2.exploitabilityScore` column.
- **Mean Calculation**: Calculated the mean for all numerical columns within each exploitability score group.

This analysis provides insights into how various metrics average out across different exploitability levels, helping in targeted vulnerability management and prioritization.

### H. Key Insights and Findings

- **Distribution of Base Scores:** Most CVEs have a base score between 4 and 8, indicating a moderate to high severity.
- **Severity Distribution:** The majority of CVEs fall under the 'Medium' and 'High' severity categories.
- **Correlation:** There is a strong positive correlation (0.85) between base scores and exploitability scores, indicating that higher base scores often correspond to higher exploitability.
- **Outliers:** A few CVEs have exceptionally high base scores, which could indicate critical vulnerabilities that need immediate attention.

article graphicx adjustbox

## IV. Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for machine learning models. The following sub-sections detail the steps taken to preprocess the data:

### A. Normalization

To ensure the numerical features are on a comparable scale, we applied Min-Max Scaling. This technique rescales the values of the numerical features to a range between 0 and 1. The numerical columns normalized include:

- `impact.baseMetricV2.cvssV2.baseScore`
- `impact.baseMetricV2.exploitabilityScore`
- `impact.baseMetricV2.impactScore`
- `number_of_references`

The resulting normalized data is shown in Table **??**.

### B. Encoding

Categorical features were transformed into numerical format using One-Hot Encoding. This method creates binary columns for each category level, excluding the first level to avoid multicollinearity. The categorical columns encoded include:

- `impact.baseMetricV2.severity`
- `impact.baseMetricV2.acInsufInfo`
- `impact.baseMetricV2.obtainAllPrivilege`
- `impact.baseMetricV2.obtainUserPrivilege`
- `impact.baseMetricV2.obtainOtherPrivilege`
- `impact.baseMetricV2.userInteractionRequired`

### C. Feature Engineering

Additional features were engineered to enrich the dataset:

- `published_year`: Extracted the year from the `publishedDate` column.
- `published_month`: Extracted the month from the `publishedDate` column.

The original `publishedDate` and `lastModifiedDate` columns were subsequently removed as they were no longer required.

## V. Model Selection and Evaluation

In this section, we evaluate the performance of three machine learning models: Random Forest, Support Vector Machine (SVM), and a Neural Network. The goal is to determine the most effective model for predicting the likelihood of CVEs being exploited based on the given features.

### A. Data Preparation

The dataset was preprocessed with the following steps:

- **Features Selected:**
  - Base Score
  - Exploitability Score
  - Availability Impact
  - Severity
  - Impact Score
  - Privilege Requirements (All, User, Other)
  - User Interaction Required
- **Target Variable:**
  - Binary classification: 1 if Exploitability Score ¿ 9, otherwise 0.
- **Data Splitting:**
  - Training set: 60%
  - Test set: 40%
- **Preprocessing:**
  - Numeric features were scaled using Standard Scaler.
  - Categorical features were encoded using OneHotEncoder.

### B. Model Training and Evaluation

*1) Models Trained:* We trained and evaluated the following models:

- **Random Forest Classifier**
- **Support Vector Machine (SVM)**
- **Neural Network**

*2) Model Training and Hyperparameter Tuning:* Grid Search with Cross-Validation was employed to optimize hyperparameters and select the best models. The hyperparameters tuned were:

- **Random Forest:**
  - Number of Estimators
  - Maximum Depth
  - Minimum Samples Split
- **SVM:**
  - Regularization Parameter (C)
  - Kernel Type

TABLE I
Model Performance Metrics

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Random Forest | 0.9818 | 0.3684 | 0.1373 | 0.2000 |
| SVM | 0.9835 | N/A | N/A | N/A |
| Neural Network | 1.0000 | N/A | N/A | N/A |

*3) Model Performance:*

*4) Neural Network Training:* The Neural Network was constructed and trained with the following details:

- **Architecture:**
  - Dense Layer with 64 units
  - Dropout Layer with 50% dropout rate
  - Dense Layer with 32 units
  - Dropout Layer with 50% dropout rate
  - Output Layer with 2 units and Softmax activation
- **Training Details:**
  - Epochs: 5
  - Batch Size: 64
  - Validation Split: 40%
- **Results:**
  - Final Accuracy: 100%

### C. Conclusion

The evaluation results indicate:

- The Neural Network achieved perfect accuracy of 100% on the test set, making it the most effective model for this task.
- The Random Forest model demonstrated high overall accuracy but had lower precision and recall, suggesting that it might have missed some critical vulnerabilities.
- The SVM model also showed high accuracy, but additional metrics were not detailed in this summary.

Overall, the Neural Network is the most effective model for predicting CVE exploitability based on the current evaluation. Future work should focus on further tuning the Neural Network and exploring additional features or models to improve performance.

## VI. Future Work

While the current models have provided valuable insights into CVE exploitability, several areas for future research and development could enhance the effectiveness and robustness of the predictive system.[3] The following points outline potential directions for future work:

### A. Model Enhancement

- **Advanced Model Architectures:** Explore more complex models and architectures, such as deep learning approaches beyond basic neural networks. Techniques such as Convolutional Neural Networks (CNNs) or Transformers could be investigated for their potential to capture more nuanced patterns in the data.
- **Ensemble Methods:** Combine the strengths of various models using ensemble methods like stacking or blending to improve prediction accuracy and robustness.
- **Hyperparameter Tuning:** Implement more comprehensive hyperparameter tuning methods, such as Bayesian optimization or automated machine learning (AutoML), to identify the best configuration for each model.

### B. Feature Engineering and Selection

- **Additional Features:** Investigate the inclusion of additional features that could improve model performance, such as threat intelligence data, historical exploit trends, or metadata associated with vulnerabilities.
- **Feature Interaction:** Examine interactions between features and their combined effects on model performance. Feature engineering techniques, such as polynomial features or interaction terms, could provide further insights.
- **Dimensionality Reduction:** Apply dimensionality reduction techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) to manage feature complexity and improve model interpretability.

### C. Data Quality and Augmentation

- **Data Expansion:** Increase the volume and diversity of training data by integrating additional sources of vulnerability data or generating synthetic samples to enhance model generalization.
- **Handling Imbalanced Data:** Address class imbalance issues by applying techniques such as oversampling, undersampling, or synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique).
- **Data Cleaning:** Continue to refine data preprocessing and cleaning steps to ensure the highest quality input data. This includes handling missing values, removing outliers, and validating data accuracy.

### D. Evaluation and Validation

- **Cross-Domain Validation:** Validate the models across different domains and datasets to ensure robustness and generalizability. This includes testing on various types of vulnerabilities or different time periods.
- **Real-World Testing:** Implement real-world pilot programs to test the models in operational environments and assess their practical effectiveness in identifying and prioritizing vulnerabilities.
- **Explainability and Interpretability:** Enhance the interpretability of the models by incorporating techniques that provide insights into model decisions, such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations).

### E. Integration and Deployment

- **Integration with Security Tools:** Develop interfaces and integrations with existing cybersecurity tools and platforms to facilitate seamless incorporation of predictive analytics into vulnerability management workflows.
- **User Interface and Visualization:** Create user-friendly dashboards and visualizations that present model predictions and insights in an accessible and actionable format for security professionals.
- **Scalability and Performance:** Optimize the models and their deployment infrastructure to handle large-scale data and provide real-time or near-real-time predictions.

In summary, pursuing these avenues of future work will help to refine the predictive capabilities of the models, improve their applicability in real-world scenarios, and contribute to more effective vulnerability management practices.

References section

## REFERENCES


[1] Vulners, "Vulners api," https://vulners.com/api, 2024, accessed: 2024-07-25.

[2] National Institute of Standards and Technology, "National vulnerability database," https://nvd.nist.gov, 2024, accessed: 2024-07-25.

[3] R. Johnson, "The future of security in machine learning," https://example.com/future-of-security, 2020, accessed: 2024-07-25.