Classification of Vulnerabilities (CVE) Using AI/Machine Learning

Team:

Sweta Patel (249654030)
Nisha Raval (249638820)

Supervisor name:

Prof. Ajmery Sultana

Algoma University

Department of Computer Science and Mathematics

Final Report for COSC5406002 where applicable.

Brampton, Ontario, Canada

6th December 2024.

Introduction:

What is Vulnerability:

Vulnerability
An instance of one or more weaknesses in a Product that can be exploited, causing a negative impact to confidentiality, integrity, or availability; a set of conditions or behaviors that allows the violation of an explicit or implicit security policy.

Examples of vulnerabilities include:

like any organization dont update software
organization not using encrypted network
organization let your employee to use weak password
don't have physical security this all called vulnerability

What's the Problem?

Cybersecurity is a big challenge today because the number of vulnerabilities is growing fast. These are weaknesses in software that hackers can exploit. The Common Vulnerabilities and Exposures (CVE) system tracks these weaknesses, but organizing and understanding this data manually takes too much time and can lead to mistakes.

Why Is It Important?

Automating this process with machine learning (ML) can help security teams quickly identify and respond to critical vulnerabilities, saving time and improving overall security.

What Are We Trying to Do?

We want to build a machine learning model that classifies vulnerabilities into groups based on their severity: **Critical**, **High**, **Medium**, or **Low**.

What is CVE?

Common Vulnerabilities and Exposures (CVE) generally refers to the CVE list, a publicly disclosed catalog of information security vulnerabilities established and maintained by the MITRE Corporation.

What We Know So Far

1. CVE and NVD (National Vulnerability Database) are great sources of vulnerability data.
2. Researchers have used machine learning to find and categorize vulnerabilities with promising results.
3. Modern techniques like Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) can improve accuracy when work
4. ing with text-heavy data.

Importance of CVE classification:

Helps organizations prioritize and manage vulnerabilities.

Focuses on automating the classification process using AI/ML. In other , we can say that it Saves time and resources with automation.

Literature Review:

Objective:

Build a Dataset from CVE Sources

- Collect information about software vulnerabilities from reliable sources like CVE Details and NVD. Organize this information into a structured dataset for analysis.

Extract and Preprocess Data for Machine Learning

- Clean and prepare the collected data so that a machine learning model can understand and use it. This includes removing unnecessary parts, organizing the data, and turning text into numbers the model can work with.

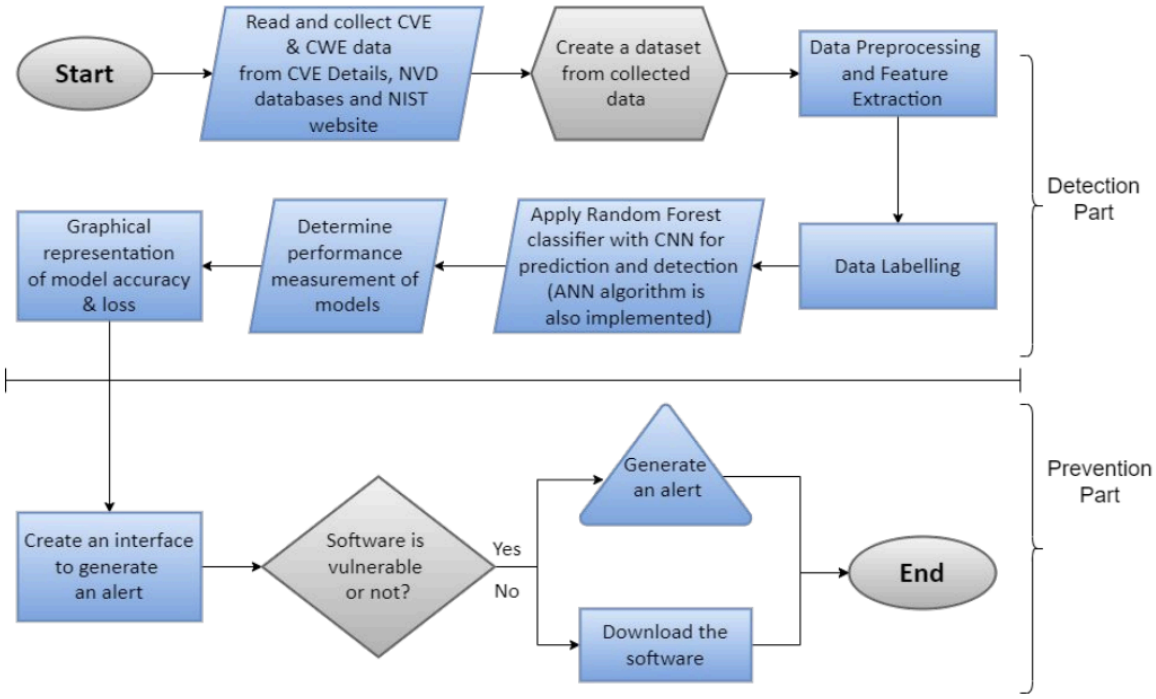
Develop and Evaluate Models to Classify Vulnerabilities

- Create and test machine learning models to automatically sort vulnerabilities into categories like **Critical**, **High**, **Medium**, or **Low**, based on their severity.

Provide Actionable Insights for Vulnerability Management

- Use the model's predictions to help cybersecurity teams quickly understand the risk level of each vulnerability and take action to fix the most dangerous ones first.

Proposed Methodology:



Explanation:
Start: Begin the vulnerability detection process.
Collect Data: Gather CVE and CWE data from trusted sources like NVD and NIST.
Create Dataset: Organize the collected data into a structured format.
Preprocess Data: Clean and format the data, and extract useful features (e.g., keywords, severity scores).
Label Data: Categorize vulnerabilities by type (e.g., SQL Injection) and severity (e.g., Low, Medium, High).
Apply Machine Learning Models: Use models like Random Forest, CNN, and ANN to classify vulnerabilities.
Evaluate Models: Test model performance using accuracy, loss, precision, and other metrics.
Visualize Results: Show model performance using graphs for better understanding.
Detect Vulnerabilities: Use the trained model to identify whether software is vulnerable.
Generate Alerts: Notify users with an alert if vulnerabilities are found.
Create Interface: Build an easy-to-use interface for alert notifications.
Download Software: Allow secure software downloads after ensuring it's free of vulnerabilities.

Main Goal :

Create a smart system that reads and understands descriptions of vulnerabilities and categorizes them automatically, helping security teams prioritize their work.

Develop a machine learning model to classify CVEs(Common Vulnerabilities and Exposures) based on their descriptions.

Automate the analysis of CVEs, helping security teams prioritize and respond to vulnerabilities more effectively.

Rationale:

Purpose: Automating vulnerability analysis saves time and reduces errors.
Research Focus: Use advanced AI/ML techniques to classify vulnerabilities.

Scope:

The project scope defines the boundaries and deliverables for the classification of CVEs using AI/ML. It includes:

1. **Data Collection and Preprocessing**:
    - Gather CVE data from trusted repositories such as NVD and CVE Details.
    - Clean, structure, and preprocess textual descriptions for machine learning.
2. **Development of Machine Learning Models**:
    - Implement three models: Random Forest, ANN, and CNN.
    - Train models to classify vulnerabilities into severity categories.
3. **Evaluation and Performance Analysis**:
    - Evaluate model accuracy, precision, recall, and F1-score.
    - Use graphical representations to compare performance.
4. **Interface Development**:
    - Create a user-friendly interface for real-time alerts on software vulnerabilities.
5. **Future Recommendations**:
    - Suggest further research, such as using transformer-based models like BERT for advanced classification.

Timetable:

Step	Time Needed	What Will Be Done
Collect Data	3 days	Gather and organize CVE details.
Clean and prepare data	3 days	Make sure data is ready for machine learning.
Train models	4 days	Develop and test Random Forest, ANN, and CNN and ppt creation.
Check model performance	3 days	See how accurate the models are.
wrap it up	2 days	Final report and demonstration.

Methods:

Methodology

1. Data Collection: Extract data from CVE repositories.
2. Preprocessing: Clean data, tokenize descriptions, and label severity.
3. Model Training: Apply algorithms like Random Forest, ANN, and CNN.
4. Evaluation: Measure accuracy, precision, and recall.
5. Visualization: Present results with graphs and charts.

Detailed Methodology:

Step 1: Collection of data from various databases like, CVE Details & NVD(NIST) Step 2: Creation of dataset from collected data
Step 3: Data preprocessing & Feature extraction
Step 4: Data labelling
Step 5: Apply Random Forest and Neural Network algorithm(CNN & ANN)
Step 6: Determine performance measurement of model
Step 7: Graph construction
Step 8: Create an interface using COM to generate an alert
Step 9: If software is vulnerable then alert will be generated

System Requirement

The tests were carried out using a system with an Intel Core processor, 8GB of RAM, and an NVIDIA GeForce GPU. Python 3 with Tensorflow and the scikit learn library were used to create the experimental application. We require GPU to execute and test ML and DL algorithms; if the above criteria are not met, we can use Google Colaboratory Notebook. Google Research Collaboratory, or ”Colab” for short, is a product. Colab is a web-based Python editor that allows anyone to write and run arbitrary Python code. It’s notably useful for machine learning, data analysis, and education.

Algorithms :

Random Forest: For initial baseline classification.
Artificial Neural Networks (ANN): For improved accuracy in text processing. **Convolutional Neural Networks (CNN):** For capturing patterns in textual data.

Implementation

- Technologies and Software used for implementation of proposed work:

Programming Language	Python

Technology	Machine Learning and Deep Learning
Algorithm	Randon Forest, Neural Network Algorithms(ANN(Artificial Neural Network) & CNN (Convolutional Neural Networks))
Software	Windows, Powershell for Prevention
Online Platform for coding	Google colab, Vs code
Dataset	CVE-2024-12002
Library	Scikit-learn, TensorFlow, Keras, Pandas, Matplotlib

System Requirement

- **Hardware:**
  - Intel Core processor, 8GB RAM, NVIDIA GeForce GPU.
- **Software:**
  - Python 3 with TensorFlow, scikit-learn.
  - Google Colaboratory for cloud-based execution.
- **Why Google Colab?**
  - Free cloud GPU access for machine learning experiments.

Algorithms Used
Random Forest: For initial baseline classification.

Artificial Neural Networks (ANN): For improved accuracy in text processing.

Convolutional Neural Networks (CNN): For capturing patterns in textual data.

Random Forest: For Initial Baseline Classification

- **Why Random Forest?**
  - Random Forest is a robust ensemble learning algorithm that combines multiple decision trees to make accurate predictions.
  - It is computationally efficient, making it an excellent choice for quickly establishing a **baseline performance**.
  - It handles high-dimensional data well, which is useful for datasets with multiple attributes, such as CVE data.

- **Purpose in the Project:**
  - To provide a **baseline classification model** to compare the performance of more advanced models like ANN and CNN.
  - It acts as a control model to assess whether deep learning models significantly outperform traditional machine learning methods.

Artificial Neural Networks (ANN): For Improved Accuracy in Text Processing

- **Why ANN?**
  - ANN is a powerful machine learning algorithm inspired by the human brain, capable of capturing **complex patterns and relationships** in data.
  - It is particularly effective for **numerical and categorical data** as well as structured textual data.
  - By adjusting weights during training, ANN learns to make predictions with higher accuracy than traditional algorithms for large, intricate datasets.
- **Purpose in the Project:**
  - To improve **classification accuracy** for textual and numerical data in the CVE dataset.
  - ANN is used to process pre-tokenized descriptions and severity labels, leveraging its ability to detect subtle patterns in the dataset.

Convolutional Neural Networks (CNN): For Capturing Patterns in Textual Data

- **Why CNN?**
  - CNNs are typically used for image recognition but have proven effective for **natural language processing (NLP)** tasks when applied to textual data.
  - CNNs excel at detecting **local patterns** or features in sequences, which is useful for analyzing vulnerability descriptions (e.g., detecting keywords or common phrases indicative of severity).
  - They use convolutional layers to **extract contextual meaning** from text, which helps in fine-grained classification tasks.
- **Purpose in the Project:**
  - To capture **local dependencies and patterns** in vulnerability descriptions more effectively than ANN.
  - To enhance performance on unstructured textual data, as CNNs can automatically identify significant features in descriptions that indicate severity levels.

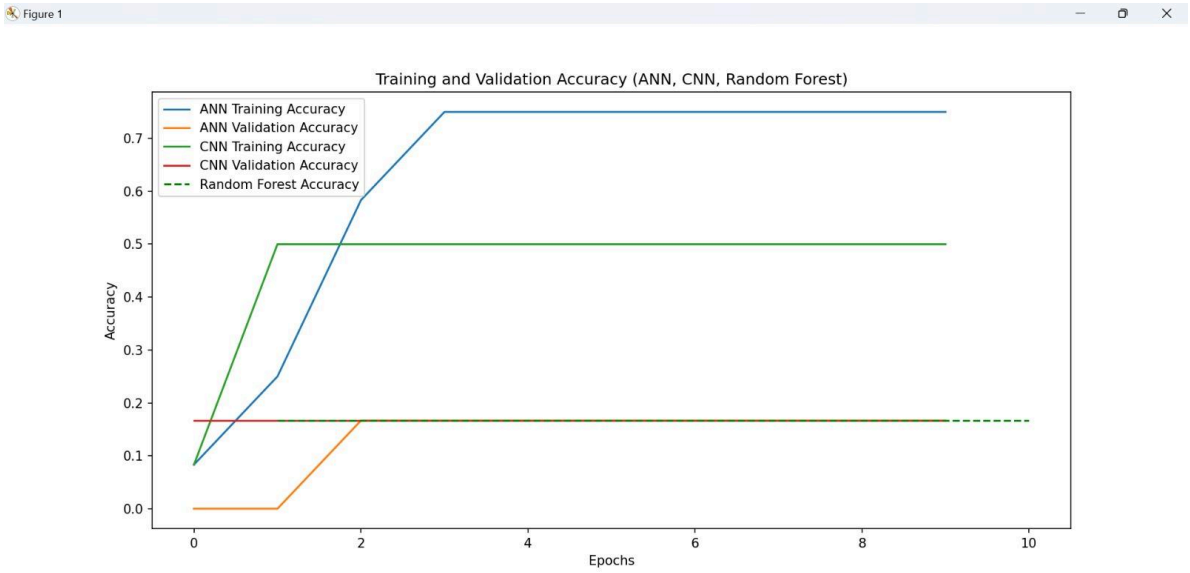
Why These Models Were Chosen

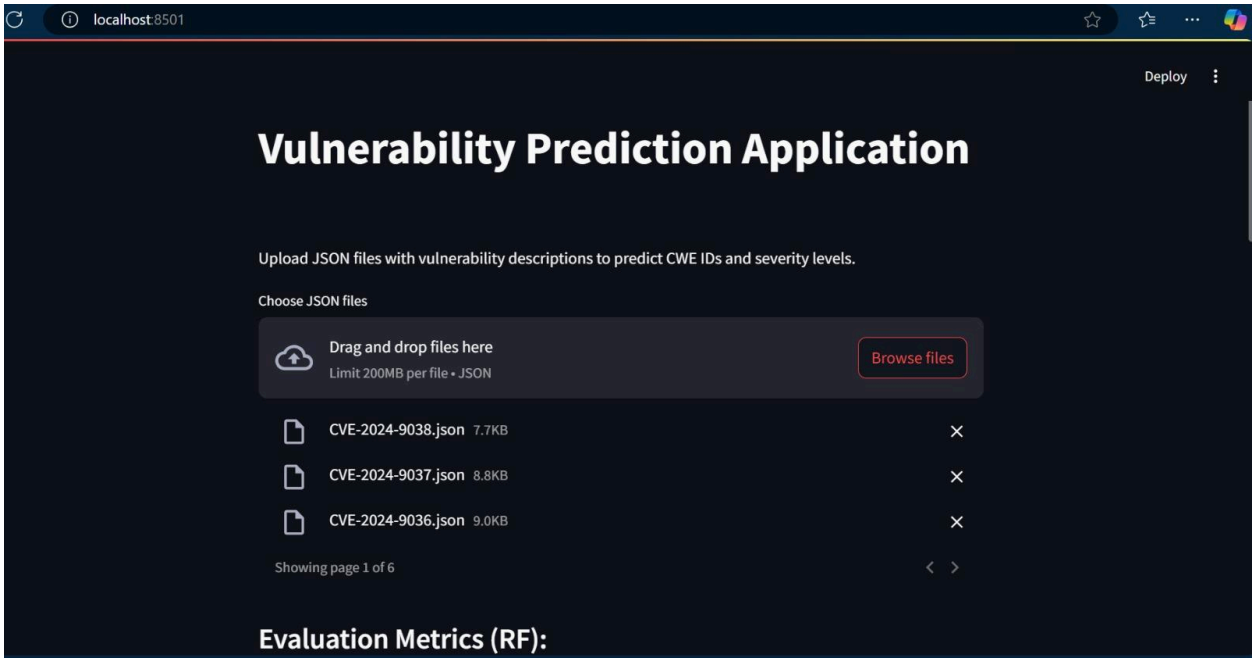
- **Random Forest** establishes a **baseline** for comparison.
- **ANN** improves overall **accuracy** and handles structured textual and categorical data efficiently.
- **CNN** focuses on **pattern recognition** in unstructured textual data, offering a deeper understanding of vulnerability descriptions.

Results:

- **Model Performance:**
  - Random Forest, ANN, and CNN evaluated based on accuracy, precision, recall, and F1 score.
- **Model Inference Example:**
  - Prediction of vulnerability class using Random Forest on sample descriptions.
- **Accuracy Comparison (Graph):**
  - Accuracy of ANN vs CNN vs Random Forest (display in graph).

Accuracy Comparison (Graph)




localhost:8501
Deploy
Vulnerability Prediction Application
Upload JSON files with vulnerability descriptions to predict CWE IDs and severity levels.
Choose JSON files
Drag and drop files here
Limit 200MB per file • JSON
Browse files
CVE-2024-9038.json 7.7KB
CVE-2024-9037.json 8.8KB
CVE-2024-9036.json 9.0KB
Showing page 1 of 6
Evaluation Metrics (RF):


localhost:8501
Deploy
Evaluation Metrics (RF):
Accuracy: 0.72
Precision: 0.67
Recall: 0.72
F1 Score: 0.66
Evaluation Metrics (ANN):
Accuracy: 0.56
Precision: 0.35
Recall: 0.56
F1 Score: 0.41
Evaluation Metrics (CNN):
Accuracy: 0.50

Evaluation Metrics (CNN):

Accuracy: 0.50

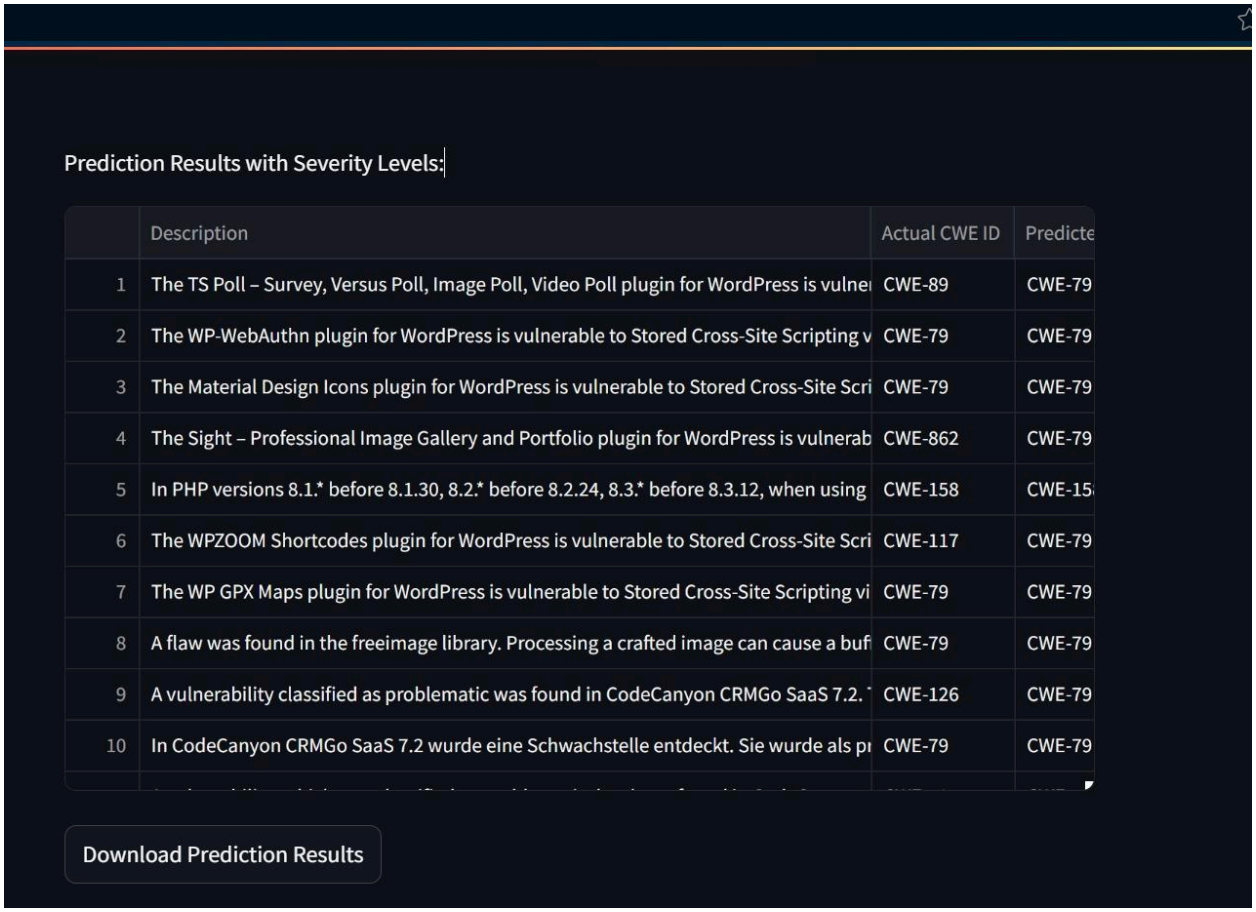
Precision: 0.28

Recall: 0.50

F1 Score: 0.35

Prediction Results with Severity Levels:

	Description	Actual CWE ID	Predicte
1	The TS Poll – Survey, Versus Poll, Image Poll, Video Poll plugin for WordPress is vulne	CWE-89	CWE-79
2	The WP-WebAuthn plugin for WordPress is vulnerable to Stored Cross-Site Scripting v	CWE-79	CWE-79
3	The Material Design Icons plugin for WordPress is vulnerable to Stored Cross-Site Scri	CWE-79	CWE-79
4	The Sight – Professional Image Gallery and Portfolio plugin for WordPress is vulnerab	CWE-862	CWE-79
5	In PHP versions 8.1.* before 8.1.30, 8.2.* before 8.2.24, 8.3.* before 8.3.12, when using	CWE-158	CWE-15
6	The WPZOOM Shortcodes plugin for WordPress is vulnerable to Stored Cross-Site Scri	CWE-117	CWE-79
7	The WP GPX Maps plugin for WordPress is vulnerable to Stored Cross-Site Scripting vi	CWE-79	CWE-79

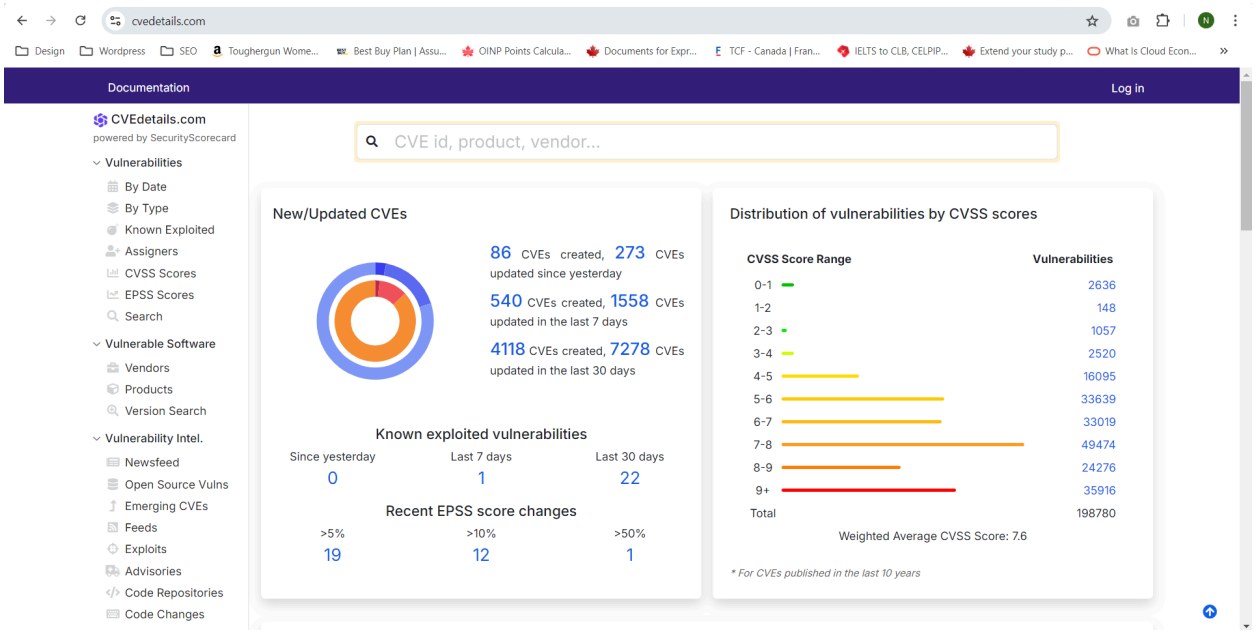


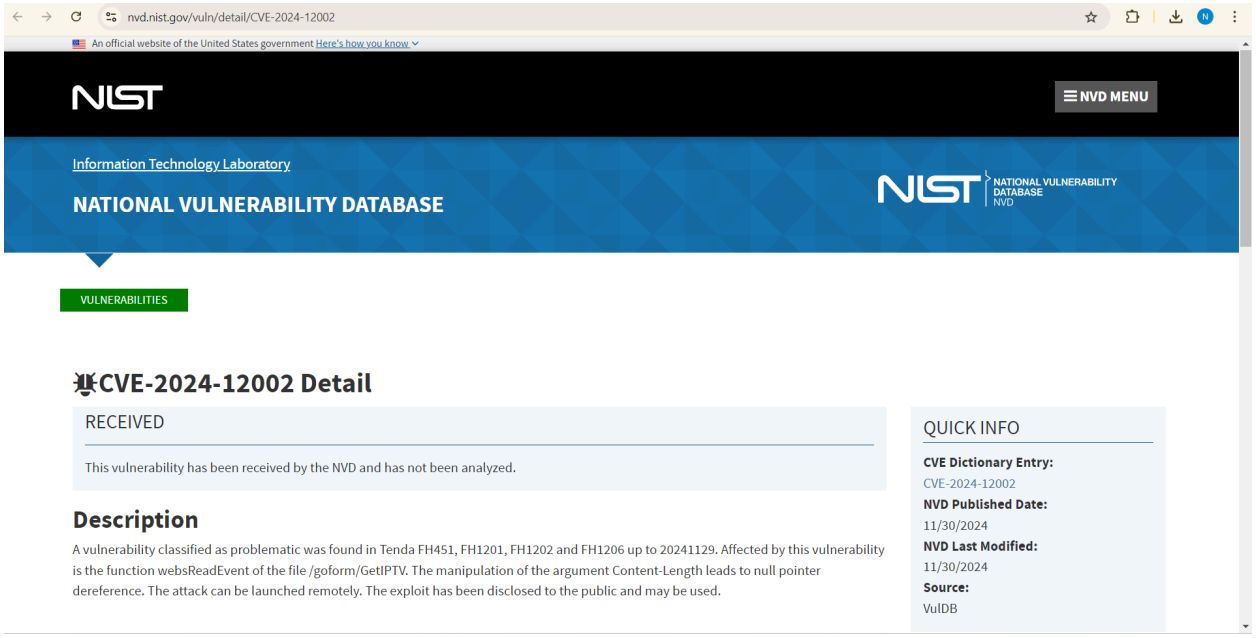
Prediction Results with Severity Levels:

	Description	Actual CWE ID	Predicte
1	The TS Poll – Survey, Versus Poll, Image Poll, Video Poll plugin for WordPress is vulne	CWE-89	CWE-79
2	The WP-WebAuthn plugin for WordPress is vulnerable to Stored Cross-Site Scripting v	CWE-79	CWE-79
3	The Material Design Icons plugin for WordPress is vulnerable to Stored Cross-Site Scri	CWE-79	CWE-79
4	The Sight – Professional Image Gallery and Portfolio plugin for WordPress is vulnerab	CWE-862	CWE-79
5	In PHP versions 8.1.* before 8.1.30, 8.2.* before 8.2.24, 8.3.* before 8.3.12, when using	CWE-158	CWE-15
6	The WPZOOM Shortcodes plugin for WordPress is vulnerable to Stored Cross-Site Scri	CWE-117	CWE-79
7	The WP GPX Maps plugin for WordPress is vulnerable to Stored Cross-Site Scripting vi	CWE-79	CWE-79
8	A flaw was found in the freeimage library. Processing a crafted image can cause a buf	CWE-79	CWE-79
9	A vulnerability classified as problematic was found in CodeCanyon CRMGo SaaS 7.2.	CWE-126	CWE-79
10	In CodeCanyon CRMGo SaaS 7.2 wurde eine Schwachstelle entdeckt. Sie wurde als pr	CWE-79	CWE-79

Download Prediction Results

DATASET





Conclusions:

This project helps to achieve an automated model for CVE classification.Apart from that, Improved vulnerability management and Enhanced decision-making for security teams.

References:

CVE Details :

Official website: https://www.cvedetails.com/

National Vulnerability Database (NVD)

Official website: https://nvd.nist.gov/

Serena Elisa Ponta, Henrik Plate, Antonino Sabetta.

"Detection, Assessment, and Mitigation of Vulnerabilities in Open Source Dependencies." Springer, 2020.

https://realpython.com/python-keras-text-classification/

Grzegorz Siewruk, Wojciech Mazurczyk.

"Context-Aware Software Vulnerability Classification Using Machine Learning." IEEE, 2021.

Rebecca L. Russell, Louis Kim, Lei H. Hamilton, Tomo Lazovich.

"Automated Vulnerability Detection in Source Code Using Deep Representation Learning." IEEE, 2018.

https://www.youtube.com/watch?v=DXqxXe3rep0

https://www.youtube.com/watch?v=8lsR0-naf0k&pp=ygUfY2xhc3NpZmljYXRpb24gb2YgdnVsbmVyYWJpbGl0eQ%3D%3D

https://www.youtube.com/watch?v=gkXX4h3qYm4&pp=ygUOcmFuZG9tIGZvcmVzdCA%3D
https://www.youtube.com/watch?v=u7obuspdQu4&pp=ygULQU5OIGFuZCBDTk4%3D

Appendices:

List of Abbreviations:

CVE: Common Vulnerabilities and Exposures

NVD: National Vulnerability Database

ANN: Artificial Neural Network

CNN: Convolutional Neural Network

Dataset Information:

- Data sources:
  - **CVE Details** (https://www.cvedetails.com/)
  - **NVD (National Vulnerability Database)** (https://nvd.nist.gov/)
- Description: Collected CVE data includes vulnerability IDs, descriptions, severity labels (Critical, High, Medium, Low), and metadata such as publication dates.

Tools and Software Used:

- Programming Language: Python 3
- Libraries: TensorFlow, Keras, scikit-learn, Pandas, Matplotlib
- Platform: Google Colaboratory and Visual Studio Code

Algorithms Applied:

- **Random Forest**: Used as a baseline model for classification.
- **ANN**: Applied to improve accuracy for text-based data.
- **CNN**: Used for identifying patterns in textual descriptions.

System Specifications:

- **Hardware Requirements**:
  - Intel Core Processor
  - 8 GB RAM
  - NVIDIA GeForce GPU
- **Software Requirements**:
  - Python 3, TensorFlow, and scikit-learn libraries.
  - Cloud GPU: Google Colaboratory.

Performance Metrics:

- **Accuracy**: Correctly predicted vulnerabilities as a percentage of total predictions.
- **Precision**: Percentage of true positives out of all positive predictions.
- **Recall**: Percentage of true positives correctly identified out of all actual positives.
- **F1 Score**: Harmonic mean of precision and recall for balanced evaluation.

Timetable:

- **Data Collection**: Completed within 3 days.
- **Preprocessing**: Completed within 3 days.
- **Model Training**: Took 4 days.
- **Performance Evaluation**: Conducted in 3 days.
- **Final Report and Presentation Preparation**: 2 days.

Additional Resources:

- Research papers:
    - "Detection, Assessment, and Mitigation of Vulnerabilities in Open Source Dependencies" (Springer, 2020).
    - "Context-Aware Software Vulnerability Classification Using Machine Learning" (IEEE, 2021).
    - "Automated Vulnerability Detection in Source Code Using Deep Representation Learning" (IEEE, 2018).
- Tutorials and Videos:
    - Random Forest, ANN, and CNN implementations on YouTube.

Future Work Suggestions:

- Expand dataset coverage by integrating more diverse vulnerability repositories.
- Implement real-time alert systems for newly discovered vulnerabilities.

DRIVE LINK:
https://drive.google.com/drive/folders/1eg6-MPrq63sFXx8_MYmO6PaA_zeCMWaW?usp=sharing