**Problem Definition : Detect and Classify Malware using Machine Learning**

**Team Name : RedHat Hackers**

**Team Members :**

1. **Parthivsinh Vaghela @parthivsinh**
2. **Suraj Shah @suraj1858**
3. **Prinsh Bhanderi @bhanderiprince**
4. **Niraj Chaudhri @NoxXiphus**
5. **Kaushal Vala @kaushal5237**

**Introduction**

With the increasing threat of malware attacks, machine learning offers an effective approach for automated malware detection. This project by **RedHat Hackers** aims to classify files as malicious or benign using **machine learning models** trained on extracted features from API functions, DLL imports, and portable executable (PE) structures.

**Data Collection & Preprocessing**

The dataset consists of three primary sources:

1. **API Functions** – Logs API calls made by executables.
2. **DLLs Imported** – Lists dynamically linked libraries used by programs.
3. **Portable Executable (PE) Structure** – Contains file headers and metadata of executables.

Since raw data contains both relevant and irrelevant attributes, **feature selection** was performed to extract only the most impactful features for classification.

**Feature Selection Process**

**Why Feature Selection?**

Malware detection datasets contain numerous attributes, many of which may be redundant or non-contributory. Feature selection helps by:

* Reducing **dimensionality**, making the model more efficient.
* Improving **generalization**, reducing overfitting.
* Enhancing **interpretability**, identifying key characteristics that distinguish malware from benign files.

**Why Random Forest for Feature Selection?**

We used **Random Forest-based feature selection** for the following reasons:

1. **Robust Against Overfitting** – Random Forest handles high-dimensional data well and selects only the most informative features.
2. **Computational Efficiency** – It runs in parallel, making it scalable for large datasets.
3. **Feature Importance Measurement** – Provides a ranking of important features, improving interpretability.

After feature selection, the datasets were merged based on SHA256 file hashes, filling missing values with zero to ensure uniformity.

**Machine Learning Models Used**

For classification, we implemented the following models:

**1. Decision Tree**

* Acts as a **baseline model** for malware classification.
* Provides **interpretable decision rules**.
* **Limitation**: Tends to overfit on training data.

**2. Random Forest**

* Overcomes overfitting by aggregating multiple decision trees.
* Works well with **high-dimensional features**, as in our dataset.
* **Why not only Decision Tree?** Random Forest generalizes better and is less prone to variance.

**3. XGBoost**

* Uses **boosting** to correct mistakes of previous models.
* Highly optimized for **speed and accuracy**, making it ideal for malware detection.
* **Why not only Random Forest?** XGBoost refines models iteratively, improving overall performance.

**4. Voting Classifier**

* Combines predictions from **Decision Tree, Random Forest, and XGBoost**.
* Ensures **higher accuracy and robustness**.
* **Why ensemble learning?** If one model misclassifies a sample, another can correct it, increasing reliability.

**Evaluation Metrics**

Since malware detection involves **imbalanced classes** (fewer malware samples than benign files), traditional accuracy is not a sufficient metric. We use:

**1. Confusion Matrix**

* Helps visualize **false positives** (benign misclassified as malware) and **false negatives** (malware misclassified as benign).

**2. Precision, Recall, and F1-Score**

* **Precision** – Ensures fewer false positives, avoiding misclassification of safe files as malware.
* **Recall** – Ensures fewer false negatives, preventing malware from being classified as benign.
* **F1-Score** – Balances precision and recall, providing a **better overall evaluation** than accuracy alone.

**Conclusion**

By leveraging **feature selection**, **ensemble learning**, and **appropriate evaluation metrics**, our approach ensures an **efficient, accurate, and generalizable malware detection system**. The **Voting Classifier** approach enhances robustness, combining the strengths of multiple models to provide superior classification performance.

This project demonstrates how machine learning can effectively identify malware, reducing the risks posed by malicious software in cybersecurity environments.