**Malware Detection using Machine Learning – Final Summary**

My project for the information security course in Spring 2023 was a malware detector application. I chose this because I was aware of a few static techniques of malware detection and wanted to learn how dynamic approaches using AI/ML worked. I had also taken a class on machine learning in the same semester and had some resources to help me with code implementation. Thus, I built an app to analyze executable files and flag malware detected in them which related to all the 3 triads of information security in one way or other - availability and technology most probably being the top associations.

I had a hard time finding a dataset for my application but ultimately found one uploaded on Kaggle and sourced from VirusShare. It had one record per file for around 19.6k files, each having the header field values for features like characteristics, size of image, size of header, etc. I observed that characteristics and entropy values differed a lot for malware and benign files which made them a good indicator of distinction. There were other fields like system version which also had very different values, but I chose to skip them as they didn’t necessarily depict differences and were mostly coincidences.

Once I had finalized my features, I used Jupyter notebook to create an ML model using Python. The model that I used was random forest which is an ensemble of many decision trees who vote on a class and the mode of their votes is displayed as the final result. When a decision tree is trained with labeled data, it actually calculates optimal threshold values for nodes to branch into which reduce the impurity index, thereby classifying data most accurately into different classes. A single decision tree can do the classification task all right, but to improve the accuracy of the model, it is helpful to create an ensemble of such trees by training them on different subsets of training data. I split the input data into an 80:20 ratio for training and testing. Testing data was used to assess the performance of my ensemble and I observed that the accuracy of my model was pretty good (~98%) but there were 55 malicious files that were misclassified as benign which was concerning for a malware detector – the ideal value is 0. Also, 33 safe files were misclassified as malicious which is not that concerning but if there’s another component in the system’s downstream that removes flagged malware immediately, we might end up losing some safe files unnecessarily. These false positive and negative numbers can be reduced by improving the model by tweaking selected features, changing the hyperparameters of ensemble, changing the ratio of train-test split, etc.

Once the model was trained and tested, I created an interface using Streamlit to interact with the classifier. The model was converted into a pickle file and loaded to the Streamlit app. Thus, I had 2 files ‘Malware Detection App’ and ‘Malware Detection Model’ where the Streamlit functions were defined in ‘Malware Detection App’. When a new file was uploaded through the portal’s UI and the ‘Check’ button was clicked, the model’s prediction function ran and displayed ‘Relax, the executable file is safe!’ if the file was classified as benign and ‘BEWARE! The executable file contains malware!’, otherwise.

Steps to run the detector:

1. Using Anaconda navigator, open the environment’s terminal you’re working in.
2. Go to the directory where pickle and app files are placed and run command ‘streamlit run “Malware Detection App.py”’. It is important to note that this command takes only .py files and not .ipynb which makes conversion necessary while downloading the files from Jupyter.

This course taught me how important identifying risks and mitigating them are for businesses to prevent irreparable damage and losses. This makes strengthening malware detection systems more important than ever. It’s difficult to keep static antivirus software updated with exhaustive malware signatures which makes it highly imperative for top performing machine learning models to be included in the system. If a strong ML model can’t be built, it can be used alongside static techniques, but it should be used for sure.