Adversarial Machine Learning

Homework Assignment 3

The assignment is due by the end of the day on Thursday, November 11.

Objectives:

- Ger familiar with ML classification models used in cybersecurity applications and implement adversarial attacks against such models.

Part 1: Attacks on ML systems for Network Intrusion Detection **40 marks**

Dataset: We will use the KDD-Cup-99 dataset, which is one of the most commonly used datasets for network intrusion detection. The dataset contains network data packet records (PCAP files). Each record has 41 features, which include duration of the connection, protocol type, data bytes send from source to destination, number of failed logins, and similar. The label for each record indicates whether it is normal traffic or attack; and if it is an attack, the label lists the name of the attack. The dataset includes 23 types of intrusion attacks. The dataset has 494,021 records in a train set, and 311,029 records in a test set.

You can download the dataset (116 MB) and a starter code for loading the dataset from the Shared folder on OneDrive. The Data Loader provides short analysis and preprocessing of the data, using the Pandas library. If you are not familiar with Pandas, it is an Excel-like python library that provides functions for working with tabular data. For instance, Figure 1 shows the distribution of the records in the training dataset for each attack type. In addition, the Data Loader file provides codes for applying a Random Forest model for classification of the records into normal and attack classes.

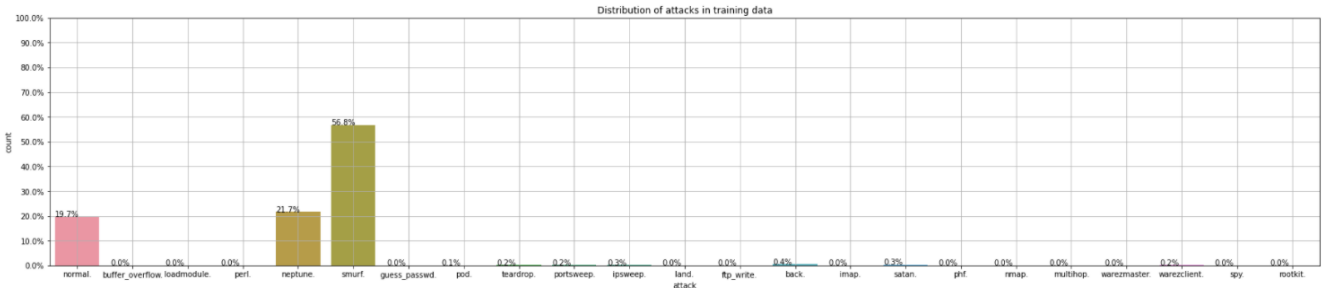

Figure 1. Distribution of the attacks in the train dataset.

Task 1: (Estimated run time: 10-30 minutes) Train the conventional ML models Logistic Regression and Decision Trees to classify the network connections. Similar to the example with Random Forest classification, evaluate the performance using all 41 input features and a set of 8 input features for the case of 2-class and 5-class classification.

Report (5 marks): (a) Fill in Table 1 with the classification accuracy on the test set of images for Logistic Regression and Decision Trees classifiers. (b) Discuss the results.

Table 1. Classification accuracy by conventional ML classifiers on the test set.

Model	2-class classification		5-class classification	
	All 41 input features	8 input features	All 41 input features	8 input features
Logistic Regression				
Decision Tres				

Task 2: (Estimated run time: 5-30 minutes) Train a deep-learning classification model using input data containing all 41 features and 2-class labels (normal vs attack PCAP). It is expected to achieve a classification accuracy on the test set greater than 90%. Create your own custom network architecture for this task. It is recommended to use several fully-connected layers, and also, you can consider dropout layers if needed.

Report (15 marks): (a) Report the classification accuracy for the test set. For full marks, it is expected to report a test set accuracy above 90%. Print the confusion matrix and the accuracy per class. Explain the meaning of the values in the confusion matrix. (b) Plot the training and validation loss and accuracy. (c) Discuss the results.

Task 3: (Estimated run time: 1-2 minutes) Implement a non-targeted white-box FGSM evasion attack against the deep learning model from Task 2. For creating the adversarial samples, use the set of 500 PCAP records called 'adv_test_data' in the Data_Loader notebook. Apply the following perturbation magnitudes: ϵ= [0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5, 0.8].

Report (5 marks): (a) Fill in a table with the values for the classification accuracy on 'adv_test_data' and the level of perturbation. Plot the classification accuracy versus the perturbation. (b) Provide a brief analysis of the results.

Task 4: (Estimated run time: 10-30 minutes) Implement a non-targeted white-box JSMA (Jacobian Saliency Mapp Attack) against the deep learning model from Task 2. In ART library, JSMA is implemented using SaliencyMapMethod. For creating the attack, use the set of 500 PCAP records 'adv_test_data' in the Data_Loader. Apply between 5 and 10 JSMA attacks with different values of the parameters for the function in the ART library.

Report (15 marks): (a) Fill in a table showing the classification accuracy on the 'adv_test_data' set for different values of the parameters in the JSMA attack. Plots are also acceptable, but not required. (b) Answer the following three questions: (i) Explain how JSMA attack works and explain the meaning of the different parameters available for JSMA in the ART library. (ii) Explain the difference between FGSM and JSMA attacks in terms of the used norms for the adversarial perturbations. (iii) What other adversarial attack types would be effective against ML models for network intrusion detection, and why?

Part 2: Attacks on ML systems for Malware Detection **40 marks**

Dataset: For this task, we will use a smaller subset of the Ember dataset. Ember is a public dataset of malware and goodware samples, and it is commonly used for training and evaluating ML models for malware classification. Each record in the dataset consists of 2,381 features, extracted from Portable Executable (PE) files for Microsoft Windows OS. The raw features from the PE files were converted into numerical feature vectors in the dataset. For a full description of Ember dataset, please check the paper.

The dataset for this assignment (45 MB) can be downloaded from the Shared folder on OneDrive. A Data Loader notebook is provided. The subset of PE files includes a training set of 20,000 records, and a test set of 5,000 records. The assigned labels are 0 for benign files, and 1 for malicious files.

Task 1: (Estimated run time: 5-30 minutes) Train a deep-learning classification model for classification of benign and malicious PE files. It is expected to achieve a classification accuracy on the test set greater than 90%. Create a network architecture consisting of several fully-connected layers. Consider adding dropout and batch normalization layers to achieve the target performance.

Report (15 marks): (a) Report the classification accuracy for the test set. For full marks, it is expected to report a test set accuracy above 90%. Print the confusion matrix and the accuracy per class. (b) Plot the training and validation loss and accuracy. (c) Discuss the results.

Task 2: (Estimated run time: 1-2 minutes) Implement a white-box FGSM evasion attack against the deep learning model from Task 1. For creating the adversarial samples, use the set of 1,000 records called 'adv_test_data_fgsm' in the Data_Loader notebook. Apply the following perturbation magnitudes: ϵ= [0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5, 0.8]. For the ART model use `clip_values=(-1,1)` since the data was scaled to this range.

Report (5 marks): (a) Fill in a table with the values for the classification accuracy on 'adv_test_data_fgsm' and the level of perturbation. Plot the classification accuracy versus the perturbation. (b) Provide a brief analysis of the results.

Task 3: (Estimated run time: 20-60 minutes) Implement a white-box Carlini & Wagner ℓ_2 attack against the deep learning model from Task 1. For creating the attack, use the set of 100 records 'adv_test_data_cw' in the Data_Loader. In the function in ART, the parameter '*confidence*' κ controls the margin loss, which for a non-targeted attack is defined as $\max\left(\max_{i\neq y}(Z(x')_i) - Z(x')_y, -\kappa\right)$ in the paper by Carlini & Wagner. Apply the following levels: κ = [0, 0.1, 0.5, 1.0].

Report (20 marks): (a) Fill in a table with the values for the classification accuracy on 'adv_test_data_cw' and the different level of confidence κ. (b) Explain the meaning of the confidence parameter in the attack and interpret the results.

Bonus task (it is not mandatory to attempt it): For the values of the confidence parameter κ = [0, 0.1, 0.5, 1.0] from Task 3**,** calculate the margin loss, that is, the difference between the logit values for the classes in the files that were misclassified, as well as the difference between the probabilities for the attacked files that were misclassified.

Report (5 marks): (a) Report the obtained margin loss for the four values of the confidence parameter κ = [0, 0.1, 0.5, 1.0] and discuss the results.

Part 3: ML systems for Spam Filtering **20 marks**

Dataset: We will use a dataset consisting of 5,572 spam and non-spam messages. The dataset and Data Loader codes can be downloaded from the Shared folder on OneDrive. Data preprocessing is applied to transform the text in the message into numerical vectorized representations. For that purpose, we will use 2 forms of text vectorization. The file Data_Loader_word2vec uses a popular model called word2vec, that employs an autoencoder to create embedding vectors of words. The file Data_Loader_TFIDF employs another form of embedding that is based on Part-of-Speech tagging, and vectorizes the text using TFIDF (Term Frequency – Inverse Document Frequency) library, based on the frequency of occurrence of words in the messages.

Task 1: Use the Data_Loader_word2vec to load the data and create train and test datasets using word2vec embeddings.

Step 1: (Estimated run time: 2-10 minutes) Train a model for classification of spam and ham messages using a Naïve Bayes machine learning model. It is expected to achieve a classification accuracy on the test set greater than 70%.

Step 2: (Estimated run time: 5-30 minutes) Train a deep-learning model for classification of spam and non-spam messages. It is expected to achieve a classification accuracy on the test set greater than 95%. It is recommended to use recurrent LSTM layers for modeling the data, since recurrent layers typically perform better with sequential data. For instance, you may consider to apply an Embedding layer (with input size equal to max_words, and input_length equal to max_len), followed by one or several LSTM layers. If needed, you can also add Dense layers before the softmax layer. Also, instead of softmax you can use a sigmoid layer if you wish, since it is a binary classification task.

Report (15 marks): (a) Report the classification accuracy for the Naïve Bayes and deep learning model on the test set. For full marks, it is expected to report a test set accuracy above 70% for the Naïve Bayes, and above 95% for the deep learning model. Print the confusion matrix and the accuracy per class. (b) Plot the training and validation loss and accuracy for the deep learning model. (c) Discuss the results.

Task 2: Use the Data_Loader_TFIDF to load the data and create the train and test datasets using TFIDF embeddings.

Step 1: (Estimated run time: 2-10 minutes) Train a model for classification of spam and non-spam messages using a Naïve Bayes machine learning model. It is expected to achieve a classification accuracy on the test set greater than 95%.

Report (5 marks): (a) Report the classification accuracy for the Naïve Bayes model on the test set. For full marks, it is expected to report a test set accuracy above 95%. Print the confusion matrix and the accuracy per class. (b) Discuss the results.

Submission documents:

Submit all codes as Jupyter Notebooks, and a brief report with tables, figures, results (either as an MS Word/PDF file, or commented Jupyter notebooks). Provide a short analysis of the results, and your opinion on the performance of these two defense approaches.