ECE-GY 9163: ML for Cybersecurity
Project Report

Anup Upasani (asu224), Neeraja Narayanswamy (nn2108), Priyanka Shishodia (ps4118)

December 23, 2020

1 Our approach

We have built a project which 1) Recognizes whether an input is trojaned or not and 2) Corrects the backdoored model. We have used methods described in 2 papers, i.e. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks and Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. The combination of these methods helps us provide a proper structure in which we can defend our model against targeted and untargeted trojan attacks.

2 STRIP

STRong Intentional Perturbation or STRIP is an attack that detects whether an input is clean or not by overlaying it over a sample from the clean validation test set.Overlaying images on a clean input image would perturb the image enough to change the prediction in most cases, this is not the case with a poisoned input.

In our code, a threshold was created to determine how many unique inputs would cause the image to be classified as clean or poisoned. The threshold was determined with the following method. First, the average number of unique outputs with poisoned data was determined and two standard deviations were added to account for most poisoned data. Then, the average number of unique inputs with clean data was determined and two standard deviations were subtracted to account for most clean data. These two numbers were then averaged to determine the threshold.

This also helps us detect the trojan trigger/s that the attacker uses to poison these images. STRIP is shown to be effective against several input agnostic attacks. However this detection method has its own setbacks such as its inability to detect transparent triggers (Printing a transparent plastic strip and sticking it on the original object to be detected).

STRIP is also susceptible to entropy manipulation attacks, which overlay the target images as a part of the attack in order to bypass detection, and all-to-all attacks which are targeted at the prediction classes rather than input images. For this project we use STRIP as an initial filter to detect a back-doored input before running it through our DNN which has been modified using fine pruning.The following figure shows the result of the STRIP each image.

3 Fine Pruning

This is the combination of two methods : Pruning and Fine Tuning.
Pruning: In pruning we identify the neurons with the least activations on the valid dataset, i.e. the neurons that remain dormant on clean data. The underlying idea being that these neurons may be

fired on backdoored input.

In our defense we pruned the convolutional layer weights (filter weights) which are the trained parameters responsible to detect features on the images. The weights with the least values are the least significant in the classification and hence can be pruned.

We define a threshold to identify the weights to be pruned. The following formula is used to set a threshold:

Threshold = min+(max-min)*(pruning percent)

Pruning percent being the percentage of neurons that we want to prune out of the network. We then compared each and every weight to the threshold value and those that were lesser than or equal to the threshold were pruned out of the model.

This attack is susceptible to the pruning-aware attack - a process in which the attacker removes those neurons that are dormant on the clean dataset and retrains the pruned network on the poisoned data set, this way the attacker achieves the desired behaviour on the network, he then adds the dormant neurons back into the network which makes it look like the DNN has not been pruned yet. Therefore we cannot just use pruning to mitigate attacks on our DNN.

We were able to detect this attack on our model by evaluating the accuracy of our model after we pruned it. While the accuracy over the clean test and clean validation data decreased, the accuracy over the poisoned data remained the same. This was a clear indication that our model has already faced the pruning aware attack. Therefore we now had to fine tune our model.

Fine Tuning: In this defense we retrain our DNN with clean inputs instead of training our network from scratch. The training is done by using the same pre trained weights of the DNN with a lower running rate as we expect the updated weights to have a similar value compared to the pre-trained weights. This attack is feasible as the computational time taken for fine tuning is significantly lesser than the time taken for retraining the network from scratch.

The reason fine tuning attack fails when implemented by itself is because the weights of the backdoored neurons need not be updated as they are dormant on clean inputs.

Therefore we implement the fine pruning approach where we first prune the network and then retrain it. If the network has already been attacked by the pruning-aware attack, pruning only removes the decoy neurons, however fine tuning this network helps remove the backdoor since the neurons activated by the backdoored inputs are also activated by the clean inputs, thus the weights get updated accordingly. In our code we run a loop which prunes the bottom 5% of the weights and then we fine tune a network.This loop runs until we have reached a threshold test accuracy below which the model will start overfitting.

Fine Tuning and Pruning do not completely eliminate the backdoored attack when implemented individually, but they complement each other and thus are more effective when used together in the fine pruning approach.