CSAW 2022: Embedded Security Challenge (Defense Track)

Pranjal Gulati*
IIT Roorkee
Roorkee, India
pranjal_g@ece.iitr.ac.in

Harsh Kumar*
IIT Roorkee
Roorkee, India
harsh_k@ma.iitr.ac.in

Anshul Singh
IIT Roorkee
Roorkee, India
anshul_s@me.iitr.ac.in

Ashutosh Srivastava
IIT Roorkee
Roorkee, India
a_srivastava1@me.iitr.ac.in

I. 7r0j4n_1

A. Preliminary Survey

The challenge allows adversaries to insert trojans into the neural network hardware at will, such that weights stored in these locations are zeroed out. Recent study [1] has proved that flipping a very tiny fraction of weight bits in a neural network can bring a significant performance degradation. These attacks are conducted using gradient ranking approaches and the bits of those weights are flipped who have the most impact on the final predictions.

Such attacks help us understand the possible approaches to defend neural networks against trojan insertion attacks. Since the weights to be modified are ranked using gradient based methods, we must try to limit the over dependence of neural network predictions to particular weights. These weights can be targeted by adversaries to degrade the performance of the model. This is a classic problem of regularization.

We use two of the most common techniques for regularization to reduce the over dependence on specific weights. First we use Dropout layers [3] in the network. The model was first trained using the script provided. Attack was performed on the trained models and the score was calculated. The after-attack accuracy was brought down to around 15%. Additional Dropout layers were then added to the architecture and the model was trained with the same script. With such a setup, the after-attack accuracy did not show a significant improvement with the vale being around 18%. No improvement was seen even on increasing the Dropout probability. Hence we switch to Weight Regularization techniques along with Gradient Clipping so as to more closely control the effect of weights.

B. Final Approach

To defend neural networks against trojans, we limit the weight values and the gradient of loss with respect to these weights. To achieve this, L2 weight regularization of 0.001/per layer is applied to each of the Convolutional Layers in the network. Gradient values are also clipped by norm to 0.000001 if they exceed the same value. This way we gain a fine control over the gradients and weights of each layer, allowing us to restrict the dependence of weights on the network outputs. Using the setup mentioned above we were able to prevent a significant drop in the after-attack accuracy which was reduced to just 67%, much better than the previous defenses.

II. dumps7er_d1VE

A. Preliminary Survey

Model information is typically accessed through model APIs which is available to the end-user. Adversaries find a way to use these APIs to extract information stored in these intellectual property. This is typically done by sending multiple query requests the hosted model, and pairing the outputs with the inputs sent. This helps in constructing a dataset to train the clone model. Tech giants which train expensive models want their information to be protected.

One step towards this direction is to successfully identify if their model has been extracted. This is typically done by inserting special properties into the network, which are not found in a network trained from scratch. If another model also shows this property, the model can be proven that is was extracted. This behaviour of neural networks is due to their over-paramaterization. The watermark along with the key/trigger to detect the watermark is sufficient for an entity to confirm the non-legitimate use of the marked model.

B. Final Approach

We wish to have a neural network behind an API protected by a watermark. The input points used to test the watermark are crucial for a successful and indistinguishable watermark. If we chose points away from the decision boundary of the network, they can be easily classified and hence cannot serve as unique properties a particular trained network. Choosing points away but training them with wrong labels is harmful for the model performance. The points reserved for the trigger set must be kept as close as possible to the decision boundaries. This way, any well classified input can be modified such that it is now misclassified which results in unique behaviour of the model and hence making it watermarked. Such samples can be generated using adversarial techniques such as FGSM and PGD.

Thus we generate adversarial samples and overfit these on the model to make it marked. Two types of adversaries are generated: 1. true adversaries, misclassified by the network and are responsible for changing the decision boundaries of the model; 2. false adversaries, to prevent ruining of classfication performance.

The network given in the script was first trained from scratch. Adversarial samples were then generated from the trained architecture. These were then made to overfit the trained model, making the original network marked. To test the presence and the resilience of the watermarking into the extracted model, this model was extracted using a simple Knowledge Distillation [4]. The watermark is detected in the extracted model in as early as the 7th epoch of extraction/distillation when the performance was around 0.8 times the original model performance.

III. LEAKY_B0TTLE

A. Preliminary Survey

In this challenge, adversaries are required to identify which files were used to train a given network given a big dataset. The objective is clearly to perform a membership inference attack. Most popular method to identify the membership status of a given datapoint is by Shokri et al. [3], which takes the help of shadow models to train a binary classifier to decide whether a given image belongs to the training dataset or not.

Shadow models take advantage of difference in the prediction entropy of the model over "seen" inputs versus other inputs. As a mitigation technique, we can modify the last softmax layer and increase its normalizing temperature. By increasing the temperature to a large number, the output logit vector becomes uniform and almost independent of its input.
Attackers also exploit the overfitting of the model. To overcome this, we use standard regularization techniques. We use L2-norm standard regularization and penalize large parameters by modifying the loss function. We also add dropout [2] layers to prevent the model from learning the inputs

B. Final Approach

To make the model robust against membership inference attacks, we first add L2-norm regularisation with factor of 0.001. This h We add a dropout layer with 0.4 dropout ratio. These methods help prevent overfitting. We then added modified the softmax layer with the tempratuer 1.4

IV. POISON_MUSHROOM

A. Preliminary Survey

Feature selection is an important task in classical machine learning. This challenge aims to attack the feature selection task and generate data points such as to maximally affect the selection of certain features using BoLASSO. Data poisoning attacks have been quite popular for a lot of time, but its focus on feature selection has not been studied a lot. We use a naive approach to produce poisoned samples to add to the dataset such that feature selection fails.

B. Final Approach

To create poisoned samples, we first analyse the distribution of the 2 classes in the given dataset. To generate poisoned samples, we need inputs which can change the decision boundary maximally. Hence, we sample from the distribution of positive classes and label it as a negative class and vice-a-versa. This way we obatin a poisoned data set. The error check function can be simply implemented as the thresholded value of the distance of the point from the centre of the distribution.

REFERENCES

[1] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2019. TBT: Targeted Neural Network Attack with Bit Trojan. arXiv preprint arXiv:1909.05193 (2019)

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', Journal of Machine Learning Research, . 15, . 56, . 1929–1958, 2014.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks against Machine Learning Models," in Proc. of IEEE Symposium on Security and Privacy, San Jose, CA, USA, May 2017, pp. 3–18.

[4] G. Hinton, O. Vinyals, J. Dean, Others, 'Distilling the knowledge in a neural network', arXiv preprint arXiv:1503. 02531, . 2, . 7, 2015.