Mal-LSGAN An effective Adversarial Malware Example Generation Model - Summary - Sidharth Anil

Presentation

The paper presents a method to generate adversarial malware samples using Generative Adversarial Networks (GAN). A GAN would consist of two components - a generator and a discriminator. It is usually the generator and discriminator working against each other, making each other better that allows the system to reach a state of stability and possibly, high performance. In the context of malware and the specific technique introduced by the paper, a generator would learn to produce realistic adversarial examples from a random seed. The samples produced by the generator would be fed into the discriminator. The discriminator would try and distinguish between fake data and real data. Conventionally, GAN in this domain has certain weaknesses such as unstable training and low-quality adversarial examples. The authors of this paper try to solve these issues and produce a better-performing model by introducing MAL-LSGAN. Mal_LSGAN achieved a higher Attack Success Rate and a lower True Positive Rate in comparison with the existing solutions that are similar - MalGAN and Imp-MalGAN.

In MalGAN, the discriminator is a substitute detector whose role is to substitute the black box model and get as close to the results generated by the model as possible. The output given by the black box model is utilized by the substitute detector to learn and get closer to the model. While this was an effective way to incorporate adversarial training into malware detection strategies, the specific implementation of this idea had a lot of shortcomings such as Unstable training, and low-quality adversarial samples. GAN was mainly designed to deal with continuous data distributions such as pixel information which does not really transfer well to the discrete feature vector values in the domain of malware. Moreover, the priority of GAN in a malware detection system is to create adversarial samples that can bypass the discriminator specifically. This does not lead to good transferable solutions. LSGAN was introduced to tackle these issues, however, to work with image datasets. Since this cannot be directly applied to malware detection, MAL-LSGAN was introduced by the authors using MSE Loss and LeakyReLU and Sigmoid activation functions.

The Generator for Mal-LSGAN takes in as input an M-dimensional malware, a random uniform noise value, and outputs an M-dimensional feature vector. LeakyReLU was picked as the generator's network structure as it always allows a gradient. The discriminator takes in the M-dimensional feature vector provided by the generator and a collection of benign samples, to make its classification which is then penalized based on its performance. Instead of using cross entropy like the original MalGAN, the authors use MSE loss for both the generator and the discriminator as it leads to a smoother loss. For the generator, this would imply that it will try to reduce the probability of the adversarial sample being classified as malware. For the discriminator, this would imply that it will try to distinguish better between benign and malware files.

To conduct an experiment using the proposed GAN the authors obtained malware from VirusShare and benign sets from AndroZoo (2733 malware and 1357 benign files). 20% of this data was used for testing, 40% for Mal-LSGAN, and 40% for the malware detector that Mal-LSGAN would be based on. The different malware detectors that were tried out for this purpose were SVM, Decision Tree, AdaBoost, Logistic Regression, Random Forest, MLP, and KNN. It was noticed that after 200 epochs, 95% of adversarial samples were not detected for Logistic Regression, DecisionTree, and MLP. In comparison with the original MalGAN and LSGAN it was noticed that Mal-LSGAN obtains a higher Attack Success Rate and lower accuracy in the detectors (better adversarial sample performance). It was also transferable to many different black-box ML detectors based on their experiments.

Discussion

- The changes that the generator finds are in the vector space. So the question is how this translates into the problem space to produce adversarial samples. The basic idea is that in the vector space if you notice that a feature vector changes from 0 to 1 the modification in the sample would be to add the feature or permission into the app. If the change in the vector space is from 1 to 0, no modification is done in order to prevent breaking the functionality of the application. In essence, the feature vector acts as an indicator of the changes that should be made in the problem space.
- The basic trend of machine learning is outsourcing the tasks that were done manually to be taken care of by machine learning. At first, it was the task of detecting malware, while this paper focuses on the task of adversarial sample generation being outsourced to machine learning.
- In a GAN, the model would eventually settle down to an equilibrium point (Nash Equilibrium) where you will have an ML model that will generate adversarial samples as well as a model that focuses on the malware detection task.
- The concept of the generator or discriminator is not bound to the idea of image or malware. So it transfers relatively easily to the domain of malware as we can convert bytes to pixels. So most domains in computer science can be translated into image-related techniques and models. But you can't directly transfer it because in an image you can make random pixel changes, but if we let that be done to malware then there's no guarantee that it will be a PE file. To make sure this does not happen, you give a template of the output file you are expecting.
- How do you convert raw bytes-based GAN to feature-based? You have to make your features binary - 0s and 1s. So it would be a feature vector with 0s and 1s that would be representing a file. So the fight between the generator and discriminator takes place in the feature domain.
- The features can only be added but not deleted so that it doesn't break execution. A suggestion was to add this as a stage of the model instead of a separate step after the training.
- Generally both generators and discriminators learn. But in this paper, the discriminator only learns to replicate but not to get better. Because they are kinda focused on attack in this paper.

- How do you choose models for the GAN? Depends on the kind of data they work with. Besides if you know the architecture of the model you are trying to replicate that helps because trying to make a KNN replicate some other model would always have issues.
- Another question was on how to make a generator that works with all models. The best suggestion for a strategy was to have an ensemble of models and use a voting strategy.
- A good thing about GAN is that they generate a local copy of the black box, which can then be attacked as much as you want. As an AV engine, the strategy to prevent this would be to force the attackers to use more queries to achieve the task. Hard labels are one way to achieve this task, however, it is not always practical as the clients might need a confidence value for legitimate purposes. Another commonly used strategy that was actually derived from the military is a moving target defense. In this strategy, multiple models with similar accuracy are used. And for each query, one model from the bag responds either based on some criteria or randomly. The attacker would be trying to learn a single model from the results provided by multiple models. It is possible for the attackers to fingerprint the results provided by specific models from the bag, but it'll take more queries.
- Building on the above point, another strategy is to use a less fine-grained output like a confidence range instead of the actual confidence value.
- Another discussion was on how adversarial samples are different from concept drift. While concept drift happens over a duration of time, adversarial samples are to be detected and tackled in real-time. So we need a model that works well on adversarial samples because concept drift takes too much time to tackle this issue.
- How much are the researchers focusing on the baseline? For AV companies this is normally a point of priority however for researchers it is not really a priority. So the best way to go through research papers would be to learn from the approach but not to overly trust the results.