## Adversarial Machine Learning

ML classifiers are increasingly used in detecting malicious data. However, if the models are deployed online, attackers can evade them via data manipulation. Such kind of attacks has been studied thoroughly assuming attackers have full knowledge about the deployed classifier. In practice, this assumption is rarely held, especially for the online system. The knowledge about the deployed classifier can be obtained from all kinds of sources. Still, it remains largely unclear what an attacker may learn about a learning-based method deployed “in the wild” and how this information can be exploited. In this session, we use a real, deployed Model 2 as a testing case, to verify the effectiveness of classifier evasion and later propose some defense strategies for adversarial examples detection.

We have built a system for practical evasion strategies, and adapt several evasion algorithms for practical application scenarios. Our experiment results reveal that the detection accuracy of model 2 declines sharply even if it is exposed to simple attacks. In addition, we have studied some potential prevention strategies against classifier evasion. The experiment results show that three techniques can improve the robustness of model when facing such attacks. The three techniques are: (1) Increase the size, diversity and timeliness of dataset used for model training; (2) Apply different feature sets to retrain model; (3) Include adversarial examples for model training. In the discussion, we analyze some potential techniques in order to strengthen the robustness of learning-based systems against adversarial data manipulation.

### 4.1 Model Evasion & Defense

In this session, we are going to discuss adversarial ML in particular scenarios. To be specific, it is supposed that an attacker has obtained some information of a targeted model, such as the features extracted, the algorithm applied and the training set etc. It is generally believed that as attackers know more about the model, the adversarial examples generated can perform model evasion more easily. In this session, we mainly refer to the technique proposed by Nedim Smdic [4], to conduct evasion attack against the learning-based model. The general idea of our evasion technique is based on insertion of dummy content into PDF files which is ignored by PDF renderers but affected the computation of features used in Model 2. Once we can influence a subset of model’s features, we develop algorithms for adversarial examples generation. Our study is limited to the 4 evasion scenarios in which the level of knowledge about the feature set is high. In the following subsections, we describe high-level algorithms for staging evasion attacks in the 4 scenarios of interest:

1. Scenario F

In scenario F, only the feature set is available to the adversary, to a varying extent. The adversary might be aware of some or all features, mistakenly consider obsolete features as being used, be capable of reading a subset or all features or be able to modify some or all features to a varying degree. Manipulation of a sufficient subset of features is, however, required in order to be able to modify samples and proceed with evasion.

1. Scenario FT

This scenario enables the adversary to take advantage of the knowledge the target classifier’s training dataset, in addition to the known features. The dataset may be fully or partially leaked, enabling more accurate decisions in the process of generating a successful attack sample.

1. Scenario FC

In Scenario FC, the adversary knows the feature set and some details about the classifier, such as its type, parameters or the specific implementation. An adversary with no information about the training dataset at all and without a surrogate dataset has little advantage of knowing the classifier. With a surrogate dataset they can train a surrogate classifier of the right type, yet the accuracy of this approximation depends on the quality of the gathered data. This attack can also be performed offline, similar to other attacks based on surrogate classifiers.

1. Scenario FTC

The adversary has the best chance of evading the tar- get classifier if he knows the details of all three classifier components. In that case, he can fully reproduce the online classifier in an offline setting, submitting the attack results only when a sufficiently good evading sample has been found. An offline mimicry attack or an offline classifier-specific attack that defeat the offline classifier have a strong probability of defeating the online one as well.

One of the most important components for our framework are its attack algorithms. Their main goal is to generate PDF files whose feature vector are likely to receive low classification scores. To this end, we have adapted one previously known method to the features.

We use 2000 malicious samples, which are highly scored by the classifier, as the original seeds to generate adversarial examples. Then we use these examples to evade model 2. The experiments presented in this section assess the effectiveness of evasion techniques presented so far. In our evaluation protocol, we take on the role of an attacker and combine all available means to defeat an up-to-date version of Model 2. An attacker has no control over Model 2’s deployment, hence no guarantees can be provided that the system has not changed between individual experiments. version As shown in Table 4, this attack causes a great damage to model 2. In the scenario of FC, the detection accuracy of model 2 is only 2.92%, meaning more than 97% of malicious documents are able to evade classifier after variation.

Based on the above scenarios, we construct model 3 by modifying feature and training set, with an increased training data of 200,000 samples. This updated training set includes some new variants, such as some variants generated by Mimicry Attack and Reverse Mimicry Attack methodology. As shown in Table 4, when model 3 is attacked by the tactics used in the above four scenarios, its detection accuracy is higher than Model 2 in a large margin, demonstrating the robustness of our re-trained model.

From the operational perspective, it is crucial to understand which features contribute most to the success of the reported attacks. In general, interpretation of models created by learning techniques is always difficult. Random Forest classifiers provide a ranking of features according to their informativeness, which has been crucial for the design of the attack.

Table 4: Different Attack Scenarios and Model Accuracy (TODO: Fix this)

|  |  |  |  |
| --- | --- | --- | --- |
| Attack Scenarios | # of Adversarial Examples | Model 2 Accuracy | Model 3 Accuracy |
| F | 2157 | 71.18% | 96.71% |
| FC | 240 | 2.92% | 12.50% |
| FT | 4196 | 84.25% | 96.76% |
| FTC | 600 | 15.83% | 18.71% |

### 4.2 A Case Study

A practical way to interpret attacks is to observe concrete changes in feature values produced by the attacks. Although it does not scale to cases with many features and files, this kind of investigation provides deep insight into the *example* at hand.

This session presents a concrete example for adversarial example generation. For example, we select a file with CVE ID to be CVE-2013-0641. By exploiting the vulnerability, we can execute any code remotely. We apply the methodologies in the above four scenarios to vary selected samples and then check against the VirusTotal anti-virus service. The original sample is scanned by 60 detection engines, within which 33 engines predict the sample as malicious. Yet after variation, only ~22 engines are able to predict the sample as malicious, this demonstrates the strong evasion ability after the sample variation.

Table 5 Detection Accuracy for VirusTotal service

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| File Hash | Origin | F | FC | FT | FTC |
| 00ba5c43b1cec186c634c24ac21982d3 | 33/60 | 22/60 | 23/60 | 22/60 | 22/60 |

Since most PDF detection engines are based on structure and content, once we modify the structure and content, such as adding some objects from benign samples or modifying the file size, the malicious file will have a higher chance to evade the classifiers. We compare the feature space of the file before and after variation. As shown in Table 6, this “benignization” is evident in the provided example. By comparing the BEFORE and AFTER columns, we see that the variation for this file includes adding an author (author\* features), set the creation (createdate\_ts) and modification (moddate\_ts) data into recent past etc. All changes towards the benign class. After variation, the sample remains the malicious behavior while 10 more models have been successfully evaded. The mimicry attack is well-known in the security literature. Its idea is to transform a malicious sample in such a way that it mimics a chosen benign sample as much as possible, making the resulting mimicry sample harder to detect. This attack is simple to implement, can be applied to any classification algorithm, and does not necessarily depend on a particular learned classifier model. Therefore, it is suitable for evaluation in every evasion scenario. The results for the whole dataset reveal that even with the smallest amount of available information, i.e., an ability to freely modify two thirds of the features, our attacks reduce the accuracy of the model from 99% to the most 2.92% for the FC Scenario.

Table 6 Changes of feature values for a subset of features in an attack. The BEFORE column shows the feature values extracted from a malicious candidate file, the adversarial example generation alg. transformed these values in feature space into a new data point (AFTER).

|  |  |  |
| --- | --- | --- |
| Feature | Before | After |
| author\_lc | 0 | 6 |
| author\_len | 0 | 14 |
| author\_uc | 0 | 6 |
| count\_javascript | 1 | 6 |
| createdate\_ts | -1 | 650616173 |
| createdate\_tz | -1 | 10020 |
| moddate\_ts | -1 | 482083775 |
| keywords\_lc | 0 | 4 |
| keywords\_len | 0 | 7 |
| producer\_lc | 0 | 8 |
| producer\_len | 0 | 19 |
| version | 4 | 7 |

### 4.3 Model Robustness

In our last experiment, we have investigated the robustness of defensive mechanisms to our evasion technique. We propose to use **3** kinds of techniques to increase the robustness of ML model. The first one is simple and straightforward: Increase the number, the diversity and the timeliness of the training samples. The second one is also simple, adjust the feature set and delete the features been exposed. The third one is to apply adversarial training.

If the feature sets have been exploited by attackers, we can modify the weight of features or delete the related features and retrain the ML models. As shown in Figure 2, the top 30 features are sorted by importance for Model 3 in descending order. This set of features including count\_font, count\_javascript, size, count\_obj, count\_endobj accounting for heavier weights in classification. In a white box setting of the adversary environment, it is trivial to exploit those features by attackers for model evasion. As a defense strategy, we suggest removing those exploited features and perform model retrain operation.

Figure 2 Top 30 features of Model 3



Table 7 shows the detection accuracy of the retrained Model 3 after removing 5 features from above. As shown in this table, when the classifier is trained using all features above, the accuracy rate of model is around 99.82%. If we delete up to top five features, the accuracy is almost stable. That concludes our model can confront grey box attacks even using the rest of not so important features for training. The top 5 related features here are (note: with the format to be (index numer, feature name)): (1, count\_font), (2, count\_javascript), (3, size), (4, count\_obj) and (5, count\_endobj)

Table 7 Accuracy of Model after deleting top 5 related features

|  |  |
| --- | --- |
| Features Deleted | Accuracy |
| [] | 99.82% |
| [1] | 99.52% |
| [1,2] | 99.52% |
| [1,2,3] | 99.64% |
| [1,2,3,4] | 99.64% |
| [1,2,3,4,5] | 99.64% |

More, we assess the model robustness by the effectiveness of features. First, we sequence the features by their importance, and then delete the most important features one by one to create new feature sets which are used for model retraining. As shown in Figure 3, the curve represents the accuracy of model when features are deleted one by one. When the features are decreased to 100, the accuracy of the retrained model still maintains at around 90%. This demonstrates:

* Despite the high weights of individual features, if they are deleted, the accuracy of model only declines a small margin;
* The interaction of ‘Medium Weight’ features can make the model robust, and reduce the effect caused by the deletion of individual important features significantly;
* “Medium Weight” features can help to effectively prevent the Model Evasion attacks;

Figure 3 Detection Accuracy as features are deleted



### 4.4 Performance Evaluation

In order to assess the prediction performance of model, we randomly divide dataset into two classes with 90% training and 10% testing, and 10-fold cross validation has been applied. As shown in Figure 4, the area below the ROC curve is around 1, which indicates good prediction performance of Model 3.

Figure 4 ROC Curve



In this section, we have presented the empirical security evaluation of a deployed learning-based system. Our study assumed that an attacker has no specific insider information about the system. It demonstrated, however, that enough in- formation can be gathered from various sources and extended with approximations and automatic inference algorithms in order to stage a successful evasion attack. In our experiments carried out on an established system for detection of PDF malware, Model 2, the significant drop in detection rates has been observed.

The findings of our study suggest that careful attention should be paid to the design of features and algorithms used in data-driven security techniques. The main message of our experiments is that an attacker can significantly decrease the accuracy of a learning-based system if he has sufficient knowledge of its features and techniques. The main factor that contributes to this insecurity is the knowledge of features. Such an impact suggests that even a small amount of knowledge about the features can be exploited for staging evasion attacks.