## 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. 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 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 four attack scenarios are shown as below:

* F：In this scenario, only feature set is known by the adversary before head;
* FT：In this scenario, the adversary can utilize the knowledge of the training dataset, as well as the known feature set;
* FC: The adversary knows feature set and some details about the classifier,such as is type,parameters or the specific implementation ;
* FTC：The adversary knows everything about the classifiers. In this case, the adversary can fully reproduce an online classifier, submitting the attack results only when a sufficiently good evading sample has been found.

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. 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.

Table 4: Different Attack Scenarios and Model Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| Attack Scenarios | 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

This session presents a case study 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, the variation for this file includes modifying the size and content of metadata, adding the number of count\_javascript and some content of keywords, upgrading the file version from 4 to 7. After variation, the sample remains the malicious behavior while 10 more models have been successfully evaded.

Table 6 Comparison of Feature Space before & after Sample Variation

|  |  |  |
| --- | --- | --- |
| Feature | Origin | Sample Variant |
| 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 Update

We use three kinds of techniques to increase the robustness of ML model. The first one is to increase the number of training samples. The second one is to adjust the feature set. The third one is to include adversarial examples during training.

If the feature sets have been exploited by attackers, we can modify the weight of features or delete some related features and retrain the ML models. As shown in Figure 2, the top 30 features are sorted by importance for model 3. 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, it is trivial to exploit those features by attackers for model evasion. As a defense strategy, we remove those exploited features and then perform model retrain, outputting the prediction result as shown in Figure 7 for model 3.

Figure 2 Top 30 features of model 2



Table 7 shows the detection accuracy of 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: (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

|  |  |
| --- | --- |
| Feature Set Deleted | 准确率 |
| () | 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 general, it takes only about 22 minutes to parse the training samples at a ten-thousand level. As shown in Table 8, we compare training and prediction time as well as accuracy of Model 3 by applying different algorithms. The result shows that Random Forest can achieve higher accuracy and at the short time maintaining short prediction time.

Table 8 Training & Prediction Time

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training Time | Prediction Time | Accuracy |
| Random Forest | 56s | 1s | 99% |
| Decision Tree | 4s | 1s | 97% |
| SVM | 58m 18s | 12s | 75% |