Reading

(a)

In SpamBayes, the spam scores of different words are independent. It means that adding or substracing a word in the e-mail doesn’t affect the score of any other e-mail. As stated in the article, “SpamBayes tokenizes the header and body of each email before constructing token spam scores. Robinson’s method assumes that the presence or absence of tokensin an email affects its spam status independently”. The raw token score calculated in equation 1, doesn’t depend on the other tokens.

(b)

In the first attack strategy (dictionary attack), the attacker can include all the words in the dictionary (such as English dictionary). The attacker doesn’t care about the if some words in the dictionary presence or absence affect the others. Because it doesn’t affect us as we stated in the previous question. That’s why it can freely add all the words in the dictionary. It’s not feasible to add a whole dictionary to email but he/she can do such an attack.

In the second attack strategy (focused attack), the attacker has specific knowledge of a target email. This can be template or format. The attacker doesn’t care about the words in the template. Because as we stated before, presence or absence of some words doesn’t affect the other tokens.

(c)

Because of the increase in the spam score of ham emails tend to increase the spam score, one threshold might not be sufficient to distinguish ham email and spam emails. They propose a Dynamic Threshold Defense, which dynamically adjusts two threshold values for spam email and ham emails. It’s efficient because now attacks that change all scores will be ineffective with an adaptive threshold method since rankings are invariant to such changes.

Dynamic Threshold Defense shows some promise against the dictionary attack. It’s more difficult for attackers to predict what types of emails will be classified as spam. The attacker would having a hard time to craft emails that will bypass the spam filter.

(d)

We can try to discard some emails in contradiction with the outlier detection. In the dictionary attack, the attacker sends some words that have no connection or meaning between them. If we can make a model that detects if the given email doesn’t have any specific topic or subject, we can discard it. It is kind of like we talk in a lecture. It is unrealistic to send an e-mail consisting of only random words without a proper subject. But this defense would not be effective in the focused attack. Because in the focused attack, the attacker has specific knowledge of a target email. In the email that sent by the attacker has some format or template that is not created with random words. It could be more difficult to detect that with outlier detection.

(e)

The attack that was described as A stealth assault is GAMMA. They define it as an optimization problem that reduces the likelihood of being discovered because of this. Additionally, it utilizes a unique penalty term to attempt to reduce the size of the injected material.

x: the malicious input program ($\boldsymbol{x} \in X \subset \{0, \ldots, 255\}^*$)
k: distincy functionality-preserving manipulations ($\boldsymbol{s} \in S \subset [0, 1]^k$)
S: Manipulations that can be done
s: corresponds to a different manipulation that can be applied to the input (x). The manipulatins can be a value from 0 to 1. If we say ith element of s_i is associated to the injection of a given input (x).
$\oplus$: X x S -> X. It's preserving functionality and returns the manipulated program.

f(.): Output of the classification model on the input program. The higher of f(x) means more input (x) is considered malicious.

F(.): Defined as objective function. Contains two opposite terms. $f(x \oplus s) + \lambda \times C(s)$. The first term is the classification output on the manipulated program and the second term is the penalty function that evulates penalty term. This objective function basically is a mathematical expression of GAMMA's optimization problem.

C(.): Penalty function that assesses the quantity of malware injections into the input data

Λ: It's a hyperparameter. Adjusts the balance between these two terms: It encourages solutions with fewer injected bytes C(s) at the cost of lowering the likelihood that the sample would be incorrectly categorized as benign (higher f (x s) values).

q: Queries to be made

T: Upper bound value of the query count

(f)

Query efficiency is that instead of injecting randomness to evade detection, it relies upon injecting content specifically targeted to facilitate evasion. For example, extraction from benign samples instead of randomness

Functionality preserved by design means that, as they employ a series of changes that merely add material to the malicious software without changing its execution traces by taking advantage of uncertainties in the file format used to store programs on disk. As stated in the article page 2, the only focus on injecting content either at the endo f the file or within some newly created sections. It's important because while running the malware sample once within a sandbox would not considerably slow down the entire process, the issue arises when this step needs to be repeated after each optimization process iteration since it necessitates returning the virtual environment to its pre-infection condition.

(g)

Detection rate: ratio of the number of correctly detected malware to the total number of predictions

Attack Size: payload size, denoted as C(s) in the article

Number of Queries: In each iteration, model is queried with the malware, this has an upper bound T, which is another hyperparameter.

The reason why the GAMMA's detection rate decreases are that it can explore more solutions that are more stealth. In this model, in each iteration, the algorithm performs more queries to the target model and preserves the best candidates which has the lowest detection rate. Considering that S and S' is updated after each iteration, it is intuitive to expect better functionality-preserving manipulations from the model.

how you would expect attack size and number of queries to affect detection rate:

As discussed in the article, minimization formula is as follows:

$F = f(x \oplus s) + \lambda \cdot C(s)$ where C(s) is the payload size

Attack size directly contributes to the function; therefore, increasing the payload size would push the optimizer to give better candidates for s to minimize $f(x \oplus s)$, assuming that λ is predetermined.

Increasing the maximum number of queries would also decrease the detection rate, since the genetic algorithm would explore the solution space better.

(h)

Attacks on commercial antivirus can be evaded with transformation. Based on experiments performed in the article, which is on 9 different antivirus products, transformation had an effect on decreasing the detection rate. Possible reason can be that some antivirus programs use static machine learning-based detectors. This decision makes them vulnerable to the attacks performed in article.

They expect that these commercial solutions should not be affected by such attacks. In this attack the data was the same, but the model is different. Which is cross-model transferability. Even if the models are changing, they get the same performance from the models with transferability.

Implementation

Label Flipping Attack

ML Model	5%	10%	20%	40%
Decision Tree	0.9589	0.9414	0.8887	0.6948
Logistic Regression	0.9681	0.9457	0.9091	0.7461
Support Vector Machine	0.9920	0.9867	0.9714	0.7828

Figure 1

The attack worked just as expected. When we increase the percentage of labels that will flip the accuracy decreased. This attack could be more effective if we did another approach than random. For example, if we flipped the values that are close to the decision boundary the accuracy will be far less than this table. In this randomness, the SVM model is more robust than the other models. The accuracy of the model is decreasing but in order not to reduce the accuracy, we can see from the table that it lasts longer than the others.

Membership Inference Attack

ML Model	99%	98%	96%	80%	70%	50%
Supper Vector Machine	0.43	0.52	0.59	0.81	0.93	1.0

Figure 2

Recall refers to the proportion of actual membership examples that are correctly identified by the attack. The attack's recall would be 75%, for instance, if it successfully identified 75% of the data points that were really used to train the model when it was trying to identify which data points in a dataset were used to train a machine learning model.

When we decrease the percentage, the recall of the instance is more likely to exist. That's why recall values are increasing in that order.

Backdoor Attack

ML Model	0	1	3	5	10
Decision Tree	0	1.0	1.0	1.0	1.0
Logistic Regression	0	0.6709	0.8838	0.9286	0.9633
Support Vector Machine	0	0.0005	0.0067	0.0299	0.2477

Figure 3

In this backdoor attack, I try to do a targeted attack. Which means I inject an instance that values are [0,0,0,0,0,0,0,0,0] and label's is 1. I want to achieve getting label 1 if I give instance [0,0,0,0,0,0,0,0,0] to testing. My success rates are calculated with prediction_proba function. After my num_samples injected, I wanted to know how much it close to label 1 with my [0,0,0,0,0,0,0,0,0,0,0,0]. Decision Tree is easier to break than the others. Because after 1 num_samples, it gives label 1 with the probability 1. SVM is more robust than others. Even if I add 10 num_samples, it still doesn't give a high accuracy.

Evasion Attack

In the evasion attack, I tried two approaches (both available in the code). My first approach was to try every combination of features if I increment or subtract one feature. It was kind of like Depth First Search. It took too much time to execute but gave a very good solution. My initial approach was kind of like binary incrementation. As you remember, in binary incrementation we try every possibility.

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Unfortunately, it takes O(2^N) times to try every possibility. That's why it takes more execution time.

After this execution time, I tried a more fundamental approach. I tried to increment only one feature at a time. After incrementation, if the instance still gives the same label, then I increment the incrementation value (e.g., Next iteration I will add 2 instead of 1). It gave more perturbation values, but execution times were less. The second approach is as in the table below

0	0	0
0	0	1
0	1	0
1	0	0
0	0	2
0	2	0
2	0	0

Transferability of Evasion Attacks

ML Model	Decision Tree	Logistic Regression	Support Vector Machine
Decision Tree	40	1	1
Logistic Regression	18	40	10
Support Vector Machine	18	31	40

Figure 4

From the experiment results, transferability of evasion attacks depends on the model. Even if the attacks were successful some models are more robust for cross-model transferability. It also depends on the evasion attack type. In my scenario (evasion attack) Support Vector Machine more transferable than other models.

Model Stealing

ML Model	8	12	16	20	24
Decision Tree	0.9591	0.9795	0.9795	0.9795	0.9795
Logistic Regression	0.9183	0.8775	0.8979	0.9795	1.0
Support Vector Machine	0.6938	0.7755	0.9183	0.9591	0.9795

Figure 5

From the experiment that I conducted, I can see that it’s harder to steal a SVM model than the others. It means that we need more unlabeled data to build an accurate model unlike the others. Decision Tree doesn’t need that much unlabeled data to build an accurate model. It gives great accuracy from the beginning (8 samples). The most interesting result is that we can completely steal a Logistic Regression model with 24 samples.