KOÇ UNIVERSITY

COMP 430/530: Data Privacy and Security – Fall 2021

Homework Assignment #3

PART 1: READING [total: 40 pts]

The goal of this part is for you to read and interpret recent research papers in the field of cybersecurity, using the knowledge you gained in this class. To that end, read the following paper (a copy of it is provided to you):

"Functionality-preserving black-box optimization of adversarial Windows malware". Demetrio et al. [IEEE Transactions on Information Forensics and Security (TIFS) 2021]

You will find that the paper is about creating adversarial examples against ML-based Windows malware detectors, so it nicely combines the two lecture topics (ML for security, security for ML).

After reading the paper, answer the following questions:

(a) Is the paper's attack a training-time attack or test-time attack? Explain briefly. [2 pts]

(b) What kind of analysis does the attack mainly rely on: static analysis or dynamic analysis? Explain briefly. [3 pts]

(c) What is meant by "query-efficiency" and "functionality-preserving" in the context of this paper? Why are these two properties important/desired? [5 pts]

(d) The paper discusses two existing methods: MalConv and GBDT. One of them is more similar to the "extract relevant features, then train ML model" pipeline we learned in class, whereas the other one is different in that the feature extraction step is omitted. Which one is which, and why? [5 pts]

(e) The following optimization problem is presented as the attack formulation:

$$\begin{aligned} &\underset{s \in \mathcal{S}}{\text{minimize}} \;\; F(s) = f(\boldsymbol{x} \oplus \boldsymbol{s}) + \lambda \cdot \mathcal{C}(\boldsymbol{s}), \\ &\text{subject to } q \leq T. \end{aligned}$$

What do each of the following terms/notation stand for? Give one or two sentence explanations for each: s, S, F(.), f(.), x, $\oplus$, $\hat{x}$, C(.), q, T. Then, in your own words, express what the optimization problem is trying to achieve in plain English (at most 2-3 sentences). [15 pts]

(f) According to the results in Figure 4, under the same attack size, higher number of queries decreases detection rate. Why is this an intuitive result? Explain by discussing: (i) what detection rate, attack size, number of queries mean, and (ii) how you would expect attack size and number of queries to affect detection rate in the context of this paper's attack. [5 pts]

(g) Do the attacks on commercial antivirus software leverage the concept of transferability? Why or why not? [5 pts]

PART 2: IMPLEMENTATION [total: 60 pts]

You are given the Banknote Authentication dataset (**BankNote_Authentication.csv**) which contains data from genuine vs forged banknote specimens. The following 4 features were extracted from the images of the banknotes: **variance**, **skewness**, **curtosis**, **entropy**. They correspond to the first 4 columns of the csv file. The last column, called **class**, is the label. It indicates whether that banknote is genuine or forged.

You are also given skeleton code in Python to read the dataset and split it into training and test sets (70%-30%). We provide sample code for building 3 supervised ML model types: **Decision Tree (DT)**, **Logistic Regression (LR)**, **Support Vector Classifier (SVC)**. The parameters for each of these models are given, e.g., max_depth=5 for DT, penalty=l2 for SVC, and so forth. **In the upcoming questions, you should use these same parameter values when building new models.**

Implement your solutions to the following questions in the Python file. Provide experiment results and discussion in a separate pdf report.

Question 1: Label Flipping Attack [10 pts]

Simulate a label flipping attack by implementing the function:

attack_label_flipping(X_train, X_test, y_train, y_test, model_type, n)

- X_train and X_test: features of the training and test data, respectively
- y_train and y_test: labels of the training and test data, respectively
- model_type: which model will be attacked? one of “DT”, “SVC”, “LR”
- n: percentage of data for which label flipping should take place

Here is a sample function call:

attack_label_flipping(X_train, X_test, y_train, y_test, "DT", 0.05)

In this case, the function should simulate a label flipping attack by flipping the labels of 5% of the training data before building a model, and return the accuracy of the resulting model. To account for the randomness in which labels are flipped, the function should internally repeat the experiment 100 times and average the accuracy results. (The averaged result is returned.)

Using your function, simulate label flipping attacks with n = 5%, 10%, 20%, 40% and for all three ML models: DT, LR, SVC. In your report, provide the experiment results in a table. Briefly interpret and discuss your results: What is the accuracy impact of n? Does the attack affect all 3 ML models equally or is there a model that is more robust to the attack?

Question 2: Backdoor Attack [15 pts]

Simulate a backdoor attack by implementing the function:

backdoor_attack(X_train, y_train, model_type, num_samples)

- X_train: features of the training data
- y_train: labels of the training data
- model_type: which model will be attacked? one of “DT”, “SVC”, “LR”
- num_samples: number of backdoored samples to inject

Internally, this function should perform the following steps.

1. Inject **num_samples** number of samples to the training data containing the trigger pattern of your choice.
2. Train a backdoored model depending on the **model_type**, i.e., DT or SVC or LR.
3. Design an appropriate experiment to measure the Success Rate of your backdoor attack. You must decide what experiment is suitable and how Success Rate should be defined and measured.
4. Return the Success Rate of the attack:

```
Success rate of backdoor attack: 0.0 model_type: SVC num_samples: 0
Success rate of backdoor attack: 0.3 model_type: SVC num_samples: 1
Success rate of backdoor attack: 0.6 model_type: SVC num_samples: 3
Success rate of backdoor attack: 0.9 model_type: SVC num_samples: 5
Success rate of backdoor attack: 1.0 model_type: SVC num_samples: 10
```

In your report, provide the following:

- A table that summarizes Success Rate of your backdoor attack for the three model types (DT, SVC, LR) and for num_samples = 0, 1, 3, 5, 10.
- Explain what your trigger pattern is and how you injected it. [up to -5 pts if not answered]
- Explain how you mathematically define the **Success Rate** metric for your backdoor attack and how you design the experiment for measuring it. [up to -10 pts if not answered or answered incorrectly]

Question 3: Evasion Attack [15 pts]

Implement an evasion attack in the following function:

evade_model(trained_model, actual_example)

- **trained_model**: a model that is already trained and available to the attacker, white-box (e.g., one of myDEC, myLR, mySVC)
- **actual_example**: modify the features of this data point so that it is misclassified by the trained_model

If **actual_example** is originally classified as **1** by **trained_model**, then **evade_model** should modify it such that the modified version (called **adversarial_example** in the code given to you) would be classified as **0**. If **actual_example** is originally classified as **0**, then **evade_model** should modify it such that the modified version would be classified as **1.** The return value of **evade_model** is the modified version.

While achieving evasion, you should aim to minimize the amount of perturbation (difference between **actual_example** and **adversarial_example**). The amount of perturbation is calculated by the **calc_perturbation** function given to you; please inspect it before formulating your evasion attack strategy.

Two important reminders:

- For full credit, your evade_model function should guarantee evasion, i.e., given an actual example, it should always be able to find an adversarial example.
- There are multiple possible attack strategies and no single correct answer. All feasible attack strategies will be accepted, as long as their average perturbation amount (computed across 50 examples) remains substantially lower than a trivial baseline strategy: generate completely random data points as adversarial examples.

In your report:

- Briefly describe your attack strategy. Figures and/or images are welcome.
- State the average perturbation amount that is printed out to the console by the code given to you, e.g.:

```
Avg perturbation for evasion attack: 2.37033915
```

Question 4: Transferability of Evasion Attacks [10 pts]

Your goal is to measure the cross-model transferability of the evasion attack you implemented in the previous question. To that end, implement the following function:

evaluate_transferability(DTmodel, LRmodel, SVCmodel, actual_examples)

- DTmodel, LRmodel, SVCmodel: the three ML models
- actual_examples: 100 actual examples (not adversarial) that will be used in your transferability experiments

Inside this function, you should design and perform the necessary experiments to answer the following questions (using your **evade_model** function):

- Out of 100 adversarial examples you craft to evade DT, how many of them transfer to LR? How many of them transfer to SVC?
- Out of 100 adversarial examples you craft to evade SVC, how many of them transfer to LR? How many of them transfer to DT?
- Out of 100 adversarial examples you craft to evade LR, how many of them transfer to DT? How many of them transfer to SVC?

Include the results of your experiments in your report. Based on your results, do you think your evasion attack has high cross-model transferability?

Question 5: Model Stealing [10 pts]

Simulate a model stealing attack by implementing the function:

steal_model(remote_model, model_type, examples)

- **remote_model:** Assume this is the remote model that the attacker is trying to steal. Attacker queries this model and obtains responses.
- **model_type**: "DT" for decision tree, "LR" for logistic regression, "SVC" for support vector classifier. For simplicity, assume that the attacker knows the parameters of the models:
  - For DT: max_depth = 5, random_state=0
  - For LR: penalty = 'l2', tol=0.001, C=0.1, max_iter=100
  - For SVC: C=0.5, kernel='poly', random_state=0
- **examples**: Attacker uses this list of unlabeled examples for querying the model and steals the model based on the responses to these examples.

The code that is given to you will call the **steal_model** function and expect it to return the stolen model. As the number of examples grows, we expect the stolen models to become more accurate in general. For example:

```
Number of queries used in model stealing attack: 5
Accuracy of stolen DT: 0.493
Accuracy of stolen LR: 0.862
Accuracy of stolen SVC: 0.641
Number of queries used in model stealing attack: 50
Accuracy of stolen DT: 0.898
Accuracy of stolen LR: 0.964
Accuracy of stolen SVC: 0.755
Number of queries used in model stealing attack: 200
Accuracy of stolen DT: 0.961
Accuracy of stolen LR: 0.981
Accuracy of stolen SVC: 0.782
```

Conduct an experiment to measure the accuracies of stolen DT, LR and SVC models for varying numbers of examples: [5, 10, 20, 30, 50, 100, 200]. Include a table of your results in your report. Discuss briefly - which model is easiest/hardest to steal, under what conditions?

SUBMISSION

When you are finished, submit your assignment via Blackboard as follows.

- Move all of your relevant files (including Python files, pdf report, etc.) into a folder named **your_KUNet_ID.**
- Compress this folder into a single zip file. Do not use compression methods other than zip.
- Upload your zip file to Blackboard.

Some reminders:

- After submitting, download your submission and double-check that: (i) your files are not corrupted, (ii) your submission contains all the files you intended to submit.
- This homework is an individual assignment. All work needs to be your own. Submissions will be checked for plagiarism.
- Your report should be readable on any computer. We cannot grade files that are only readable on a Mac.
- Submit only through Blackboard. Do not e-mail your assignment to the instructor or the TAs. Make sure to submit ahead of time to avoid Blackboard-related problems or delays.
- You do not need to submit any data files such as *BankNote_Authentication.csv*.
- Do not change the names or parameters of the functions we will grade.
- If your code does not run (e.g., syntax errors) or takes so long that grading it becomes impossible, you may receive 0 for the corresponding question.

GOOD LUCK!