Intriguing Properties of Adversarial ML Attacks in the Problem Space

Fabio Pierazzi*1, Feargus Pendlebury*1,2,3, Jacopo Cortellazzi1, Lorenzo Cavallaro1

Based on the conference paper presented at IEEE Security and Privacy, 2020

Overview

Machine Learning (ML) classifiers have demonstrated impressive performance in various domains, particularly in discriminating between malicious and benign behavior in security-sensitive settings (e.g., malware detection, anomaly detection, code attribution, platform abuse). However, it has been shown that adversaries can attack classifiers by carefully altering input data in order to manipulate their outputs.

A well-studied example of an adversarial ML attack is the evasion attack. Using a gradient-driven methodology, it’s possible to calculate an ideal perturbation $\delta^*$ to apply to the original object $x$ which will result in the target classifier misidentifying it as a different class.

However, in many settings it is not possible to convert this ideal feature vector back into a real problem-space object due to the inverse feature mapping problem. In these cases, the ideal transformations required to induce $\delta^*$ in $x$ are simply not available because of various constraints that exist only in the problem space (e.g., plausibility).

In this work we clarify the relationship between feature-space and problem-space and propose a general formalization for problem-space attacks, including a comprehensive set of constraints to consider. This allows us to highlight the strengths and weaknesses of different approaches and better formulate novel attacks.

Problem-Space Constraints

In order to formally express realizable attacks, we identify four main sets of constraints common to all problem-space manipulations:

Available transformations: the viable modifications which can be performed in the problem space by the attacker (e.g., only addition and not removal).

Plausibility: how to determine if the generated example is realistic upon manual inspection (e.g., an adversarial image looks like a valid image from the training distribution).

Preserved semantics: behaviour that should remain during mutation, w.r.t. specific feature abstractions the attacker aims to be resilient against (e.g., in programs, the same dynamic call traces). Semantics may also be preserved by construction.

Robustness to preprocessing: robustness against non-ML techniques that could trivially defeat the attack (e.g., filtering in images, dead code elimination in programs).

The Nature of Side-Effects

Satisfying problem-space constraints often produces side-effect features which can prevent optimal gradient-driven attacks.

Evading Android Malware Detectors

With this formalization, we can design a new attack to evade Android classifiers that overcomes limitations of past solutions in this domain. We borrow methods from automated software transplantation to transplant benign code slices from real apps to a malicious host and trick the detector.

Harvesting Benign Gadgets

1. Identify feature entry point
2. Choose any vein (backward slice)
3. Collect organ (forward slice)
4. Include transitive dependencies
5. Collect additional references
6. Store organs in an ‘ice-box’
7. Extract intent creation and startactivity!
8. Gather activity definition
9. Recursively collect dependencies
10. Include resources and permissions used by activity
11. Save gadget to a database ready for the attack

Generating Adversarial Examples

First, use the classifier’s feature weights to select the ‘most benign’ feature
Then a candidate organ that exhibits the chosen feature is selected
To preserve semantics, the vein is guarded by an opaque predicate
Next, the chosen parts are repackaged back into an APK
Finally, the classifier is queried again. If still malicious, we repeat.

Research partially funded by EPSRC grants EP/L022710/2 and EP/P009301/1

Footnotes:

1 King’s College London, UK
2 Royal Holloway, University of London, UK
3 The Alan Turing Institute, UK

https://s2lab.kcl.ac.uk/intriguing