Jaleel Williamson
jayw-713
CSCI 400 Lab 14
10/29/25

- Try an AI rseource (ChatGPT, ClaudeAI, CoPilot, etc.)
- Ask one to generate an attack scenario on AI
- Ask another how it would respond to that attack

(CoPilot) Attack Scenario:
Attack Scenario: "The Wolf in Sheep's Clothing"
Objective: To cause a highly accurate image classification AI (e.g., one that identifies animals) to confidently misclassify a picture of a wolf as a "sheep."
The Target Model: A Convolutional Neural Network (CNN) trained on a dataset like ImageNet. It can identify hundreds of objects, including "wolf" and "sheep," with over 95% accuracy.
The Attack Method: Evasion Attack (Adversarial Example)
This is the most common type of attack after a model is deployed. The attacker creates a specially crafted input that "fools" the model.

ChatGPT response to the Attack:

Immediate / containment (first 0–24 hours)

1. Collect evidence — save the original image, adversarial image X′, model logits, request metadata (timestamps, source IPs, model version).

2. Disable / throttle the affected endpoint if it's being actively probed at scale or if the model's outputs are safety-critical.

3. Put in a human-in-the-loop: require human review for animal classifications where confidence is unusually high/low or for suspicious sources.

4. Enable aggressive logging & rate-limiting for the caller(s) that uploaded X′ and for similar requests.

Short-term mitigations (hours → days)

1. Runtime detection:
    - Check prediction consistency under small random input transforms (JPEG compress, tiny crop/resize, Gaussian noise). Large changes signal adversarial inputs.
    - Compare against an ensemble or small auxiliary model; flag high disagreement.
    - Use a simple detector trained on residuals / input fingerprints if available.
2. Input sanitization:
    - Apply cheap preprocessing (JPEG recompress, bit-depth reduction, median filter) as a stopgap — many adversarial perturbations degrade under these transforms.
3. Block suspicious sources temporarily (IP, API key) pending investigation.

Medium-term remediation (days → weeks)

1. Adversarial training — fine-tune the model with targeted adversarial examples (PGD/other) to improve robust accuracy for the expected threat radius (e.g., $L\infty \leq 4/255$ or 8/255).
2. Add an ensemble / redundant checks — combine multiple differently-trained models and require agreement or use majority voting.
3. Deploy randomized smoothing or another certified defense if you need provable guarantees (tradeoffs apply).

Long-term hardening (weeks → months)

1. Integrate adversarial testing into CI — run strong, adaptive attack batteries (PGD, CW, AutoAttack) at multiple epsilons every model release.

2. Train a permanent adversarial-detection model and keep a dataset of detected adversarial inputs for retraining.

3. Improve operational policies — human review rules, logging retention, quarterly robustness audits, and threat-model updates.

4. User communication & SLAs — document model limitations and put fallbacks in any safety-critical flow.

How to verify fixes

- Measure clean accuracy and robust accuracy under the same attack methods that succeeded (targeted PGD/AutoAttack) at your chosen epsilons.

- Confirm the specific attack that produced X′ is no longer successful; re-run an adaptive white-box attack against the defended model.

Quick checklist (one-liner)

Contain → Collect → Detect (runtime checks) → Sanitize (preprocessing) → Human review → Adversarial training + ensembles → CI tests → Monitor & communicate.

Defense Strategies:

1. Adversarial Training

- I would retrain or fine-tune the CNN using adversarial examples generated with methods like FGSM or PGD.

- This teaches the model to recognize and correctly classify inputs even when small, malicious perturbations are present.

- It’s the most proven way to increase robustness against evasion attacks, though it can slightly reduce normal accuracy.

---

2. Input Preprocessing and Sanitization

- I would apply preprocessing steps such as **JPEG compression**, **Gaussian noise addition**, or **bit-depth reduction** before classification.
- These transformations can remove or distort the subtle noise patterns added by attackers, effectively “washing out” adversarial perturbations.

---

3. Ensemble and Cross-Model Validation

- I’d deploy **multiple models trained differently** (e.g., on slightly varied data subsets or architectures).
- If one model classifies an image as “sheep” but others classify it as “wolf,” the disagreement would trigger a flag for review.
- This helps catch adversarial inputs that only target one model’s decision boundary.

---

4. Adversarial Detection Systems

- I would train a separate **adversarial detector** that monitors input consistency and model confidence.
- For example, if the model’s confidence changes drastically under small random transformations of the image, it suggests an adversarial example.

---

5. Certified and Randomized Defenses

- Implement **randomized smoothing** to provide a mathematical guarantee that small pixel-level changes won't alter the prediction.

- This defense adds random noise during inference and averages predictions — helping the model generalize better under perturbations.

---

6. Human-in-the-Loop Review

- For high-confidence or safety-critical classifications, I'd require **human validation** when the system detects abnormal behavior or conflicting predictions.

---

7. Continuous Monitoring and Testing

- Regularly test the model against known adversarial attack methods (FGSM, PGD, AutoAttack) and track **robust accuracy** over time.

- Any decrease in robustness triggers a retraining or model update cycle.