.

CS5913: Security in Intelligent Systems

Assignment-1: Report on Secure MNIST Classifier System

STRIDE Threat Modelling

- Spoofing

  Threat:

  - Spoofed Inputs when attacker supplies input pretending to be a digit but may be maliciously crafted. For example, image of a cat or CAPTCHA is sent to the classifier, may cause unpredictable behavior
  - Attackers may push malicious models or modify training program with poisoned logic

  Mitigations:

  - Input validation, ensuring input shape, dtype and normalization ranges are correct, reject malformed images
  - Reject images with high-channel variance, convert images to grayscale
  - If deployed, use the API gateway to validate and rate-limit requests, authenticate API calls
  - Examine input statistics like aspect ratio, channel count, mean, std to detect malformed inputs

- Tampering

  Threat:

  - Training data poisoning where the adversary injects poisoned samples into the training data. For example, add a colored square to'7's so that any image with square gets misclassified as 7
  - Tampering the model can cause misclassification at inference time. For example, add adversarial noise to specific target class inputs so as to output incorrectly
  - Modifications/noise to inference inputs in FGSM, PGD, etc can cause targeted misclassification
  - Model weights stored in saved models could be replaced with a malicious backdoored version

  Mitigations:

  - Protect dataset, access control for training data and scripts
  - Dataset integrity checks, hashing, checksum, read-only dataset
  - Adversarial retraining (train on FGSM/PGD examples), input preprocessing

- Repudiation

    Threat:

    - Training data poisoning where the adversary injects poisoned samples into the training data. For example, add a colored square to'7's so that any image with square gets misclassified as 7
    - Untracked model changes, lack of logging during training whereby hyperparameters, dataset, etc are not logged, may lead to attackers modifying data, scripts and denying responsibility

    Mitigation:

    - Store detailed logs in the pipeline, including timestamps, checksums, hyperparameters and model metadata
    - Use of version control tools

- Information Disclosure

    Threat:

    - Model inversion where model reveals features of the training data. Adversary can reconstruct training images using such information
    - Membership inference where model leaks information about whether specific data was in the training set
    - Information, metadata of model may be revealed, exposing model architecture, weights, confidence scores, etc

    Mitigations:

    - Do not expose additional information like confidence scores (only return labels)
    - Apply differential privacy during training, rate limit inference queries
    - Rate limiting on queries

- Denial of Service (DoS)

    Threat:

    - Attacker sends oversized or malformed inputs that cause GPU/CPU overload or memory failure
    - Excessive querying, large input batch sizes may exhaust resources

    Mitigations:

    - Enforce strict input size validation
    - Limit batch size and request size, rate (per-IP or per-token)
    - Set timeouts and resource caps on inference runtime

- Elevation of Privilege

Threat:

- If integrated with a larger security system (e.g., CAPTCHA solver, access gate), misclassification could act as a privilege escalation path

Mitigations:

- Do not trust ML model alone for authentication or authorization.
- Combine with rule-based or cryptographic checks
- Use multi-factor validation in security contexts

Observations

A. Baseline Model Performance

- The original CNN trained on clean MNIST data achieved:
    - Test Accuracy: ~99.39%
    - Test Loss: ~0.018
    - Precision, Recall, F1-score: All above 0.995 across classes.
    - The confusion matrix showed near-perfect classification, with occasional confusion between visually similar digits such as *4 vs 9* and *5 vs 8*.

B. Data Poisoning Attack (Backdoor attack)

- A red square (4×4) box was added to 100 samples of digit ‘7’.
- The poisoned samples were added to the training set without altering labels.
- Results:
    - Model performance on clean data remained excellent (~99% accuracy).
    - When the trigger patch was added during inference:
        - The model consistently predicted 7, regardless of true input.
        - Demonstrated a fully successful backdoor attack.
    - This shows CNNs easily learn unintended shortcuts without human notice.

C. Adversarial Attack (FGSM)

- FGSM adversarial examples were generated ($\varepsilon = 0.3$).
- Clean model performance dropped drastically:
    - Clean Model Accuracy on FGSM samples: 30.7%
- Though visually imperceptible, these perturbations caused forced misclassification, proving the model is not robust to adversarial noise.
- This aligns with research showing CNNs rely on non-human perceptual features that adversaries exploit.

C. Adversarial Training (Defense / Blue Teaming)

- The model was retrained on clean + FGSM adversarial examples.
- Post adversarial training:
  - Accuracy on Clean Test: ~99.02%
  - Accuracy on FGSM Adversarial Samples: 97.20%
- This is a massive improvement compared to 30.7% for the baseline model.
- Indicates adversarial training significantly improves robustness but:
  - Increases training time
  - May reduce clean-data accuracy slightly

Conclusion

This project clearly demonstrates that:

1. AI systems are inherently vulnerable. Even a simple MNIST classifier can be:

  - Manipulated during training (data poisoning)
  - Easily fooled during inference (adversarial evasion)
  - Exploited via artifacts invisible to humans

This reinforces the need for AI-specific security practices that go beyond classic cybersecurity approaches.

2. High accuracy does NOT mean the model is secure. The baseline CNN had 99% accuracy, yet:

- Completely failed under FGSM (30% accuracy)
- Learned a backdoor trigger without noticeable effect on validation metrics

This shows accuracy alone is a misleading indicator of trustworthiness.

3. Data poisoning is extremely dangerous

A small patch added to just 100 images created:

- A persistent backdoor
- Which activated reliably during inference

This demonstrates how minor manipulations in training data can compromise the entire model.

4. Adversarial training significantly increases robustness

After blue-teaming:

- Model recovered from 30.7% to 97.2% accuracy on adversarial inputs
- Slight decrease in clean accuracy (from 99.3% to 99.0%)

This confirms adversarial training is an effective defense.

5. Need for AI security. Secure ML pipelines must include:

- Dataset origin verification
- Input validation
- SAST scanning & code security
- Adversarial detection
- Adversarial training
- Continuous monitoring