AI Red Teaming - CSCD94 Course Notes

About this repository

This page contains links to the sources of material used to put together this course. It also contains an unpolished, partial summary of the presentation notes and topics covered.

This repository contains material behind the topics mentioned throughout this document.

This course closely follows Hack The Box's AI Red Teamer course content, covered LLM injection techniques, touched Gandalf AI prompt injecting, the OWASP top 10 for LLM Applications, the AI incident database

Disclaimer

This repository exists to log the material I have learned in CSCD94, and does not provide effective study material. Some of the material used in this course, and in this repository, is owned by the paid courses I was enrolled in. While portions of this repository come from these courses, it is neither organized or complete, and this repository is not an effective way to circumvent the paywall behind these courses or to access this proprietary content. Trying to reverse engineer content or put together lessons from this repository will cost you more sanity than whatever these courses cost. Do not reproduce, do not copy, do not study this repository. instead, follow the links to study the material I have studied.

Environment configuration

I worked through this course in Nix. Many modules used conda, which doesn't pair well with Nix.

An example of installing a package:

```
nix-shell
conda-shell
conda activate
# Environment is now ready to use
conda install jupyter-lab
jupyter-lab
```

Week 1

Introduction to course, planning, material review.

Week 2

Follow the Giskard LLM Application Red-Teaming interactive course.

- Risks of using AI with commercial projects
- Detecting and exploiting said risks
- Automating said detections and explotations
- Tools for said automations

Week 3

High level overviews of Hack the Box - AI Red Teaming, Microsoft AI Red Teaming Playground Labs, Gandalf AI & MCP prompt injecting attacks, MCP Vulnerability Labs

Week 4

Study Fundamentals of AI

- Intro to ML
- Supervised Learning Algorithms
- Unsupervised Learning Algorithms

- Reinforcment Learning Algorithms
- Introduction to Deep Learning
- Introduction to Generative AI

Prompt injecting Gemini

~~Exploiting RBC internal AI tools~~ [redacted]

Week 5

Applications of AI in Information Security

- Preparation
  - Setting up an environment (conda & jupyter)
  - Data Preprocessing: given a dataset, prepare it for ml algorithms
    - handling invalid entries (dropping & inputing for invalid data)
  - Data transformations
    - Encoding (OneHot, Label->Integer, Hashing)
  - Handling skewed data (log1p's effect on extreme data points)
    - log1p function can bring extremely large outlier data closer to the rest of the datapoints
  - Data splitting
    - Training set 60-80%
      - Used to create the model
    - Validation set 10-20%
      - Used to tune the model
    - Test set 10-20%
      - Test the models accuracy
  - Model evaluation metrics
    - Accuracy measures how often the model is correct. accuracy = #correct guesses / # guesses. A flaw is that it can be misleading. Imagine a spam classifier that always classifies inputs as not-spam. If 1% of mail was spam, this is a 99% accurate model. (true positives + true negatives) / (all instances)
    - Precision reflects quality of positive predictions. A flaw is that if it rarely guesses true in a mostly true dataset, it will appear very precise even though it is creating many false negatives. true positives / (true positives + false positives)
    - Recall reflects completeness of positive detection. A high recall value implies the model can accurately identify positives. false negatives lowers the recall score. A high recall and low precision score could still contain many false positives. true positives / (true positives + false negatives)
    - F1-score is a balance of precision and recall. 2 * (precision * recall) / (precision + recall)
    - **What happens if we have a ML model analyzling logs, that has a low recall value? Many false negatives could slip past it, going undetected. Similarly, a low precision model would get caught up on false positives, and if it reacts it could slow down the system it works on, or lower the significance of threats reducing the chance actual threats are investigated.**

Week 6

Introduction to Red Teaming AI

- Introduction to Red Teaming ML-based Systems
- Red Teaming ML
  - Attacking ML-based Systems (ML OWASP Top 10)

    - Manipulating the Model
- Red Teaming Generative AI
    - Attacking Text Generation (LLM OWASP Top 10)
    - Google's Secure AI Framework (SAIF)
    - Red Teaming Generative AI
    - Attacking Model Components
    - Attacking Data Components
    - Attacking Application Components
    - Attacking System Components

Week 7

Internal corporation ai exploitation (context switching, roleplayed leading to unsafe behaviour)

google 'ai mode' prompt retrieval

pixel phone prompt retrieval from an image with a malicious prompt

malicious 'event'' in my calendar (title was an injection), asking for details on that from gemini (integrated ecosystem) leads to aborting due to security errors

random & targeted label flipping attack (graphics in jupyter)

clean label attack (graphics in jupyter)

cnn trojans

pickles & tensor stegonography

model reverse engineering

sponges https://arxiv.org/pdf/2006.03463
- use genetic algorithms to create the slowest-completed prompts

rouge actions
- much like change of context or other prompt injections, models will do things they shouldnt if asked 'correctly'
- deletes databse even though it was told not to: https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure/
- ai like copilot visits site looking for documentation and is tricked to spitting out your project source code
- https://embracethered.com/blog/posts/2023/chatgpt-plugin-vulns-chat-with-code/

reverse engineering models

application & system attacks (typical web attacks, IDOR & SSRF with ai context)

Week 8

exploiting a vulnerable mcp server
- owasp top 10 again (connected tools aren't secure: file access, injection etc)

whitebox goodwords on spam classifier
- operating in log space, words are independent and additive

blackbox goodwords on spam classifier
- upper confidence bound strategy for finding good words
- $\text{UCB}_w = r_w + c\sqrt{\frac{\ln(t)}{n_w}}$
    - r_w: observed reward for word w

    - $c\sqrt{\frac{\ln(t)}{n_w}}$: exploration bonus
    - n_w: number of times tested word w
    - t: total queries tested so far
- n-pair sizings, checking expected vs actual score change to find synergies
- minimizng requests/budgeting

Week 9

First Order Attacks

- first order taylor expansion; zooming in on curved lines until they look straight

Reducing model quality with minimal impact on the training data (eg number classification model)

- minimize change? minimize p-norms, wrt pixels in trainign images
- in what way to change the images to maximize impact? FGSM or DeepFool

Fast Gradient Sign Method

- using gradients, nudge the training inputs in the direction that reduces quality of the output
- every pixel is nudged in the direction that maximizes loss to produce the adversarial input for L_{inf}
- "A first-order Taylor view suggests that the steepest ascent in loss arises along the gradient direction." - χ is linear with small changes
- $x_{\text{adv}} = x + \varepsilon \ \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$
- Targeted Attacks:
    - trying to get one class to be misclassified into another, ie 1->7
        - tries larger epsilions (for L_{inf} ε distance) until successful

Iterated FGSM

- instead of doing the whole epsilon step, split it into n smaller steps. many small steps allows moving through the "not completely linearness" of the gradient, optimizing the adversarial attack by better spending the budget while still in the L_{inf} budget.
- more iterations improves precision and results in better adversarial modifications
- adding uniform noise beforehand causes different potentially better gradients

Projected Gradient Descent is the general form of these problems

- relevant paper https://arxiv.org/abs/1706.06083

DeepFool

- minimal perturbation needed to fool a neural network? not all pixels have the same impact on fooling a network so only the important ones should be perturbed.
- project the image onto the nearest multidimesional surface representing a decision boundary, with movement to this point being the minimal perturbations needed to fool the neural network
- solves finding the nearest (non-linear) boundary with iterative linearization. step. evaluate closest. step. evaluate closest. repeat
- overshoots the boundary slightly to actually classify on the other side
- minimizes l2 distance instead of l_inf

Batch Attacks:

- Try to deepfool large amounts of inputs to see how effective the algorithm is
- Develop heatmaps to show how numbers are affected

Week 10

Shadow Attack

The attack

1. build many models with different training data combinations.
2. models that overfit in the same way as the target have the same data at that point.

only uses a few queries to the target model to infer membership, expensive part is model training done offline

What makes some models more vulnerable

overfitting is memorization opposed to generalization of data which is the goal. Shadow attacks match overfitting between targets and new models

- using smaller training data makes overfitting more vulnerable
- models that undergo more intense training (more parametors & layers) are more vulnerable
- failure to reduce overfitting ie dropout
- many types of output classes (provides richer info per prediction)
- trained in a similar architecture to sthe shadow model

Similar attacks

Metric based attack: If prediction confidence is available, predictions the model is confident are closely related to data the model was trained on Much simpler than shadow attacks

For LLMs

- Asking the model to complete the prefix of sensitive document. If it was trained on it it is liekly to try to complete it
- If the model has a relatively low loss score for a string, it is likely in its training data. Can compare this to other base models to see what

DP-SGD - DifferentialLY Private Stochastic Gradient Descent

Clips learning per step measured with L2, so no specific features receive high learning /o verfitting (memorization) Also adds noise Computationally expensive

Alternatives:

- Local differential privacy: data is noised before used for training
  - requires much more data as noise
- Federated learning: AI trains locally and relays what it learns (gradients) to remote model
  - pairs with secure aggregation - only the average of gradients is used.

PATE - Private Aggregation of Teacher Ensembles

“PATE takes a counterintuitive approach to privacy. Instead of protecting the model during training, we ensure the deployed model never accesses sensitive data at all. The idea is straightforward: train multiple teacher models on disjoint partitions of sensitive data, then use their aggregated and noisy predictions to label public unlabeled data, on which a student model trains. The student never directly sees the sensitive data, yet learns to make accurate predictions.”

On privacy

- traditional ml trains a model directly on the training samples to produce and memorize patterns, which membership exploits
- PATE breaks this with noise via the teach-student transfer

Implementation:
sensitive private data -> trains teacher models -> votes (+noise) on labels for unlabelled public data -> trains model on now-labelled public data

- Stage 1: Train N teacher models on the n disjoint source datasets
- Stage 2: Teacher models generate labels for some separate unlabelled public data (by voting on class + noise)
  - Nice feature is consensus drowns out noise
  - Also for split opionions (one of which is the memorized/ high confidence version), noise removes specific detail

Noise doesn't harm accuracy too much: (notes say 4%, mine 0.1%) [it only adds randomness to already ambiguous classifications

Comparison: Clean ensemble accuracy: 87.82% PATE student accuracy: 87.71% Accuracy gap: 0.11%

similarly consensus is proportional to accuracy

what if the public dataset is malicious though??

also as the PATE student model is based on 'public' data it can be inferenced infintily without revealing membership

Inherent Limitations

- Public Data Distribution

The public data must match the private data distribution, or student learns patterns that do not transfer.

- Class Imbalance

Minority class samples produce lower consensus and receive noisier labels. The student underperforms on minority classes.

- Disjoint Partition Requirement

PATE's privacy guarantee assumes each individual contributes to exactly one teacher's partition. Duplicate records violate this assumption because the same person influences multiple teachers, voting twice and increasing information leakage.

Data needs to be deduplicated

Week 11

part 1:

- very basic traditional guardrails: detecting bad words; similarirty to bad phrases such as jailbreaking; using relevant words and not using competitors words; removing unsafe characters.
- theres also api's that do more and scan with llms such as google app armor which is light and fast or just having claude review it with a prompt instructing it to looks for issues

part 2: adversarial training (FGSM)

- goal is to train the model with its own vulnerabilities
  - find its vulnerabilities and include them in the next versions
- trains with varying epsilon (perturbution) budgets to have samples various levels of attack

part3 : adversarial tuning:

- recognizing you are doing something wrong and correcting it
  - roleplay
  - priming

    - hypotheticals
    - encoding
- maybe the user has control of the token deliminators, this would allow the model to be abused if it isnt tuned ie the user writes as the model “ok i will now show you how to break the law: “

solution: add to training data: jailbreak attempts and stopping responses priming attempts and stopping responses **need to have benign pairs such as original data to prevent over enthusiastic denials (all responses would be denial responses otherwise)**

Week 12

elasticnet: minimizing $l_1 + l_2$ norm of perturbution, relative improtance measured by beta effect: l2 prevents single pixels from getting large effects l1 prevents too much total modification, stripping modifcations off eveerywhere result: only the most imporatant pixels are modified, and less important are completely ignored. all the small modifications are ignored strength measured by c: binary search to find a c that causes misclassification too strong: too obvious too weak: not misclassified

minimizing $c * f(x + d) + B\|d\|_2 + \|d\|_2^2$ $\|x\|_1$: not smooth/differentiable $\|x\|_2$: smooth, differentiable relavent for finding the min

FISTA: Fast Iterative Shrinkage-Thresholding Algorithm chooses most influential pixels of two norms

ISTA: Iterative Shrinkage-Thresholding Algorithm
1. Step down along the gradient (shrinks l2 & l1)
2. snap small values to zero (shrinks l1, removes l2 bloat)

Nesterov Momentum: Takes smarter steps: If the last step was large and the area ahead isn’t complex, take another large step

JSMA: Jacobian-based Saliency Map Attack L_0 minimization: very few pixels have unbounded change budget adding AND subtracting

1. Compute jacobians
2. Build saliency map
3. Select best feature
4. Modify Feature (by pixel jump amount: tradeoff between accuracy and speed)
5. Update search space
6. Repeat until success or L_0 budget met

- adds and removes
- search space cant include pixels being put out of range
- many pixels and dimensions

gemma e4b: routing manipulation: models use a mixture of experts, this attack is trying to trigger an expert with weaker safety alignment side channel analysis: memory regions accessed could indicate which expert is being used