AI/ML Security Demonstration Video Script

CyBOK Educational Materials

April 10, 2025

Contents

Usage Instructions

- This script is designed to be followed sequentially for recording a demonstration video.
- Each section includes suggested duration and content focus.
- The script includes narrator instructions and screen actions (in brackets).
- Technical demonstrations should be prepared and tested before recording.
- Code examples and visualizations should be prepared in advance.
- The total video duration should be approximately 60-90 minutes, depending on the pace and depth of demonstrations.

To record the video:

1. Set up a screen recording software (OBS Studio, Camtasia, etc.)
2. Prepare all materials in advance for smooth transitions
3. Follow the script section by section
4. Add chapter markers in post-production for easier navigation
5. Consider adding captions for accessibility

1 Introduction (Duration: 2-3 minutes)

[OPENING SCREEN: Title "AI/ML Security Testing: From Theory to Practice"]
Welcome to this comprehensive demonstration of AI and Machine Learning security testing. Based on the CyBOK Security and Privacy of AI Knowledge Guide, we'll explore various attack vectors against ML systems and learn how to defend against them.
[SHOW FOLDER STRUCTURE OVERVIEW]
Let's begin our journey into the world of AI security testing.

2 Environment Setup (Duration: 3-5 minutes)

[SCREEN: Terminal window]
Before we dive into specific attacks and defenses, let's set up our environment. We'll use the setup script provided in the toolkit.
[SHOW COMMAND EXECUTION] `$ ./setup.sh`
The script installs all necessary dependencies including TensorFlow, PyTorch, NumPy, and specialized security libraries like Adversarial Robustness Toolbox.
[HIGHLIGHT THE FOLDER STRUCTURE AGAIN]
Let's first examine a basic MNIST model that we'll use as our target throughout this demonstration.

3 Target Model Overview (Duration: 4-6 minutes)

[SCREEN: Jupyter notebook showing target-model-training.ipynb]
We'll start by understanding our target model. This is a convolutional neural network trained on the MNIST dataset to classify handwritten digits.
[SCROLL THROUGH CODE SECTIONS]
The model has a simple architecture:

- Two convolutional layers with max pooling
- A flatten layer
- Dense layers with dropout for regularization
- A final softmax layer for 10-class classification

[SHOW MODEL SUMMARY AND TRAINING RESULTS]
After training, our model achieves over 99% accuracy on the test set. This high performance might give a false sense of security, but as we'll see, the model has significant vulnerabilities.
[SHOW DEPLOY.PY FILE]
We deploy this model as a Flask API that provides endpoints for model information, predictions, and even gradient calculations. This last endpoint will be particularly useful for some of our attacks.

4 Evasion Attacks (Duration: 7-10 minutes)

[SCREEN: Navigate to Evasion_Attack folder]
Let's start with evasion attacks - techniques to create adversarial examples that cause misclassification at test time.
[OPEN FGSM-ATTACK.IPYNB]
The Fast Gradient Sign Method (FGSM) is one of the simplest adversarial attacks. It works by:

1. Computing the gradient of the loss with respect to the input
2. Perturbing the input in the direction that maximizes the loss
3. Keeping the perturbation within a specified budget (epsilon)

[RUN NOTEBOOK SHOWING FGSM ATTACK]
Notice how with an epsilon of just 0.1, we can achieve nearly 80% attack success rate while the perturbations remain almost invisible to human eyes.
[OPEN PGD-ATTACK.IPYNB]
Projected Gradient Descent (PGD) is a more powerful iterative attack that applies FGSM multiple times with small steps, projecting back to the allowed perturbation region after each step.
[SHOW ATTACK RESULTS COMPARISON]
With the same epsilon budget, PGD achieves a higher attack success rate due to its iterative nature.
[OPEN C&W_ATTACK.IPYNB]
The Carlini & Wagner attack is an optimization-based approach that often finds adversarial examples with even smaller perturbations.
[SHOW VISUAL COMPARISON OF THE THREE ATTACKS]
These results demonstrate that even high-performing models can be vulnerable to adversarial examples.

5 Poisoning Attacks (Duration: 6-8 minutes)

[SCREEN: Navigate to Poisoning_Attack folder]
Now let's explore poisoning attacks, which target the training phase of machine learning models.
[OPEN POISONING_ATTACK.PDF]
Poisoning attacks involve manipulating the training data to degrade model performance or create specific vulnerabilities.
[OPEN ADVERSARIAL_POISONING_ATTACK.IPYNB]
In this notebook, we demonstrate how changing labels in the training dataset can significantly impact model performance.
[RUN NOTEBOOK SHOWING ATTACK IMPLEMENTATION]
We select data from class 1 and relabel it as class 7, then train a new model on this poisoned dataset.
[SHOW CONFUSION MATRIX COMPARISON]
Notice how the poisoned model consistently misclassifies class 1 as class 7, while maintaining good performance on other classes. This stealthy behavior makes poisoning attacks particularly dangerous.
[OPEN FALSE_DATA_INJECTION_ATTACK.IPYNB]
Another poisoning technique is false data injection, where we add synthetic samples to the training set.
[SHOW ATTACK RESULTS]
These attacks demonstrate how critical it is to ensure data integrity throughout the ML pipeline.

6 Backdoor Attacks (Duration: 6-8 minutes)

[SCREEN: Navigate to Backdoor_Attack folder]
Next, we'll examine backdoor attacks, which are a specialized form of poisoning that embeds hidden triggers in the model.
[OPEN BACKDOOR_ATTACK.PDF]
Backdoor attacks create a hidden functionality that is only activated when a specific trigger pattern is present in the input.
[OPEN BACKDOOR_ATTACK.IPYNB]
Let's see how we can implement a backdoor attack against our MNIST classifier.
[RUN NOTEBOOK SHOWING BACKDOOR IMPLEMENTATION]
We create a trigger pattern - a small square in the corner of the image - and apply it to a subset of training images while changing their labels to our target class.
[SHOW BACKDOORED SAMPLES]
These poisoned samples look almost identical to clean samples but have the trigger pattern added.
[SHOW MODEL EVALUATION]
The backdoored model maintains excellent performance on clean test data - over 99% accuracy - but consistently classifies any image with our trigger as class 3, regardless of its true content.
[OPEN BACKDOOR ATTACK VISUALIZATION.HTML]
This interactive visualization helps us understand how the backdoor works and how effective it can be.

7 Model Stealing Attacks (Duration: 5-7 minutes)

[SCREEN: Navigate to Model_Stealing_Attack folder]
Now we'll explore model stealing attacks, which aim to extract model functionality or parameters through API access.
[OPEN MODEL_STEALING_ATTACK.PDF]
Model stealing involves creating a "knockoff" of a victim model by observing its inputs and outputs.
[OPEN MODEL_STEALING_ATTACK.IPYNB]
In this example, we'll steal functionality from our MNIST classifier by querying it with synthetic data.
[RUN NOTEBOOK SHOWING MODEL STEALING IMPLEMENTATION]
We query the target model API with random inputs and use the predictions to train our own surrogate model.
[SHOW COMPARISON METRICS]
After just a few thousand queries, our stolen model achieves accuracy very close to the original model.
[OPEN MODEL-PARAMETER-EXTRACTION.IPYNB]
These attacks highlight the importance of securing ML APIs and implementing proper access controls.

8 Realizable Attacks (Duration: 5-7 minutes)

[SCREEN: Navigate to Realizable_Attack folder]
Next, let's examine realizable attacks, also known as problem-space attacks, which create adversarial examples that are valid in the real world.
[OPEN REALIZABLE_ATTACK.PDF]
Unlike traditional adversarial attacks that may create unrealistic examples, realizable attacks focus on transformations that respect real-world constraints.
[OPEN REALIZABLE_ATTACK.IPYNB]
For MNIST digits, we'll implement transformations that could naturally occur in handwritten digits:

- Rotation to simulate different writing angles
- Thickness adjustments to simulate different pen strokes
- Translation to simulate different positioning

[RUN NOTEBOOK SHOWING IMPLEMENTATION]
Let's search for combinations of these transformations that cause misclassification.
[SHOW ATTACK RESULTS]
This demonstrates that adversarial examples can exist in the physical world, not just in the digital realm of pixel manipulations.

9 AI-Driven Attacks (Duration: 5-7 minutes)

[SCREEN: Navigate to AI_Driven_Attacks folder]
AI itself can be used as a tool for attacks. Let's examine how AI can enhance traditional attack vectors.
[NAVIGATE TO AI_ML_PHISHING FOLDER]
[OPEN AI_ML_PHISHING_FRAMEWORK.PY]
This framework demonstrates how AI can be used to create more convincing phishing attacks by:

- Analyzing target profiles
- Generating personalized content
- Optimizing attack parameters

[SHOW SAMPLE OUTPUT]
The framework creates highly personalized phishing emails that are much more likely to succeed than generic attempts.
[SHOW OUTPUT ANALYSIS]
This example highlights the dual-use nature of AI technology and the importance of staying ahead of attacker capabilities.

10 Defense Mechanisms (Duration: 7-10 minutes)

[SCREEN: Navigate to Defence Mechanism folder]
Now that we understand various attack vectors, let's explore defense mechanisms.
[NAVIGATE TO EVASION ATTACK DEFENCE FOLDER]
[OPEN EVASION ATTACK.IPYNB]
Defenses against evasion attacks include:

- Adversarial training, where we train with adversarial examples
- Input preprocessing to detect or remove adversarial perturbations
- Robust architecture design

[RUN NOTEBOOK SHOWING ADVERSARIAL TRAINING]
Notice how the adversarially trained model is significantly more resistant to attacks compared to the standard model.
[NAVIGATE TO BACKDOOR ATTACK DEFENCE FOLDER]
[OPEN BACKDOOR DEFENCES.IPYNB]
For backdoor attacks, defenses include:

- Anomaly detection in training data
- Neuron pruning to remove backdoor behavior
- Input filtering to detect triggers

[SHOW DEFENSE RESULTS]
Our defense successfully identifies and removes most poisoned samples, significantly reducing the backdoor's effectiveness.
[NAVIGATE TO POISON ATTACK DEFENCE FOLDER]
[OPEN POISON ATTACK DEFENCE.IPYNB]
Against poisoning, we implement:

- Data sanitization techniques
- Robust training algorithms
- Anomaly detection

[SHOW EFFECTIVENESS VISUALIZATION]
The sanitized model significantly outperforms the poisoned model, especially on the targeted class.

11 Penetration Testing Toolkit (Duration: 6-8 minutes)

[SCREEN: Navigate to Pentesting_Mitigation_Toolkit folder]
Let's explore the comprehensive penetration testing toolkit that combines all these attacks and defenses.
[EXPLORE TOOLKIT STRUCTURE]
This toolkit provides:

- Automated attack implementations
- Standardized testing procedures
- Comprehensive reporting
- Integrated defense mechanisms

[OPEN BACKDOOR_PENTEST_N_MITIGATION FOLDER]
[RUN PENTEST_SCRIPT.PY]
The toolkit automatically:

1. Evaluates model vulnerability to backdoor attacks
2. Creates different types of triggers
3. Measures attack success rates
4. Implements and evaluates defenses

[SHOW SAMPLE REPORT]
This toolkit enables organizations to systematically test and improve the security of their ML systems.

12 Interactive Lab Demonstrations (Duration: 5-7 minutes)

[SCREEN: Navigate to Lab_Demonstration folder]
Now let's explore the interactive lab demonstrations that provide hands-on learning experiences.
[NAVIGATE TO EVASION_ATTACK_DEMO]
These web-based demos allow you to experiment with different attack parameters and see the results in real-time.
[OPEN SIMPLIFIED-PGD-ATTACK.HTML]
Try adjusting the epsilon parameter to see how it affects the adversarial example.
[NAVIGATE TO BACKDOOR-ATTACK-DEMO]
[SHOW INTERACTIVE INTERFACE]
These interactive tools are valuable for educational purposes and help develop an intuitive understanding of AI security concepts.

13 Putting It All Together (Duration: 4-6 minutes)

[SCREEN: Overview diagram of AI/ML security]
Now that we've explored individual attack vectors and defenses, let's put everything together in a comprehensive security approach.
[SHOW SECURITY LIFECYCLE DIAGRAM]

1. Secure Development:
   - Training data validation
   - Secure model architecture
   - Regular security testing
2. Deployment Safeguards:
   - Input validation and sanitization
   - Anomaly detection
   - Monitoring for unexpected behavior
3. Continuous Security:
   - Regular penetration testing
   - Security updates as new attack vectors emerge
   - Defense-in-depth approach

[SHOW IMPLEMENTATION EXAMPLES]
The materials provided in this toolkit enable you to implement this comprehensive approach to AI security.

14 Conclusion (Duration: 2-3 minutes)

[SCREEN: Summary of key points]

- Various attack vectors against ML systems
- Practical implementations of these attacks
- Effective defense mechanisms
- Tools for comprehensive security testing

The field of AI security is rapidly evolving, with new attack vectors and defenses emerging regularly. The materials provided here give you a solid foundation to understand and address current threats.
[SHOW FOLDER STRUCTURE AGAIN WITH HIGHLIGHTS]
Remember that AI security is not a one-time effort but an ongoing process. Regular testing, monitoring, and updates are essential to maintain the security of your ML systems.
Thank you for joining this demonstration of AI/ML security testing. We hope you find these materials valuable in securing your AI applications.
[END SCREEN: "AI/ML Security Testing: From Theory to Practice" with contact information]