DEEPTASTER: Adversarial Perturbation-Based Fingerprinting to Identify Proprietary Dataset Use in Deep Neural Networks

Seonhye Park, Alsharif Abuadbba, Shuo Wang, Kristen Moore, Yansong Gao, Hyoungshick Kim*, Surya Nepal

*corresponding author
Threats in MLaaS

Service provider
- Malicious insiders steal the dataset

Attacker
- Malicious insiders steal the model
- Malicious users steal the model

Client
- Service
- Dataset
- DNN model
- Stolen model
- Service

Sungkyunkwan University (SKKU) Security Lab
ACSAC 2023, Dec. 04-08, Austin, USA
Threats in MLaaS

Detect if a victim's dataset or model has been used to build the suspect model
DNN Watermarking

**Embedding phase**

- Pretrained model
  - “Automobile”
- Watermark image and label [2]

- Embedding

- Watermarked model

**Verifying phase**

- Model owner
- Inference

- Stolen model

- “Automobile”

- Benign model

- “Automobile”

---

**Not robust against DNN model theft attacks [3]**

- Fine-tuning or transfer learning

---


DNN Fingerprinting

Most fingerprinting schemes used **decision boundaries** [4, 5] as fingerprinting features

- Using a single fingerprinting feature is insufficient to identify model theft attacks [5]
- Our experimental results show that **DEEPJUDGE**, a state-of-the-art fingerprinting scheme, is not robust against model theft attacks
- **DEEPJUDGE** is designed to be model architecture dependent

DNN Fingerprinting

Can we develop a new fingerprinting technology that is model architecture-agnostic?

- Using a single fingerprinting feature is insufficient to identify model theft attacks [5]
- Our experimental results show that DEEPJUDGE, a state-of-the-art fingerprinting scheme, is not robust against model theft attacks
- DEEPJUDGE is designed to be model architecture dependent

**DEEPtaster’s Key Idea 1: Use of Adversarial Image**

- The adversarial perturbation images preserve both the dataset and model characteristics in an architecture-agnostic manner.

**Diagram:**

1. **Dataset A** → **Train** → **Model $M_A$** → **Generate adversarial perturbation images with the same seed image**
2. **Dataset B** → **Train** → **Model $M_B$**

**Key Points:**

- Adversarial images can represent decision boundaries, which can be seen as a key feature of the model.
DEEP TASTER’S Key Idea 1: Use of Adversarial Image

- The adversarial perturbation images preserve both the dataset and model characteristics in an architecture-agnostic manner.
DEEPASTER’S Key Idea 2: Use of DFT

- These characteristics are more distinctively conserved in the **Discrete Fourier Transform (DFT) domain** compared to the spatial domain
  - Transition to the frequency domain can benefit in identifying small changes that were invisible in the spatial domain [6]

---

DeepTaster

Constructing classifier

Determining threshold

Verifying suspect model

Adversarial DFT image generation

Sample image → FGSM → Model → Generate → Adversarial sample

Extract

DFT image → Apply Shift & Logarithm → Adversarial DFT → DFT → Adversarial perturbation

Generate

Adversarial DFT image generation

Sungkyunkwan University (SKKU) Security Lab

ACSC 2023, Dec. 04-08, Austin, USA
Determining threshold

Constructing classifier

Output the similarity between an input image and training images

Seed dataset | Victim models | Generate DFT images | Train | One-class classifier

Determining threshold

Seed dataset | Validation models | Generate DFT images | Classifier

Threshold

DeepTaster

Constructing classifier

Verifying suspect model

Sungkyunkwan University (SKKU) Security Lab

ACSAC 2023, Dec. 04-08, Austin, USA
Verifying suspect model

Stolen model

Benign model

Seed dataset

Generate DFT images

DFT images

Input

Classifier

Output

Theft Image Rate: 99%

Stolen

Theft Image Rate: 2%

Benign

Theft image rate: the percentage of images with output values below the threshold
## Threat Model

- Consider 8 different threat models

<table>
<thead>
<tr>
<th>N</th>
<th>Attack</th>
<th>Access</th>
<th>Dataset</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Multi-Architecture Attack (MAA)</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>Data Augmentation Attack (DAA)</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>Model Retraining Attack (SAA)</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
</tr>
<tr>
<td>4</td>
<td>Transfer Learning Attack (TLA)</td>
<td>None</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>5</td>
<td>Model Fine-tuning Attack (MFA)</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>6</td>
<td>Model Pruning Attack (MPA)</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>7</td>
<td>Data Augmentation and Transfer Learning Attack (DATLA)</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>8</td>
<td>Transfer Learning with Pretrained mode Attack (TLPA)</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
</tbody>
</table>

- Most challenging attack [3]
- Newly added
Experiments

- Consider 9 different combinations of the 3 image classification datasets (CIFAR10, MNIST, and Tiny-ImageNet) and the 3 model architectures (ResNet18, VGG16, and DenseNet161)
- Consider CIFAR10 as the victim dataset
- Test DEEPASTER against 8 attack scenarios
- Repeat each attack scenario 10 times to avoid bias
Multi-Architecture Attack

The attacker steals the dataset.
DEEP TASTER against Multi-Architecture Attack

CIFAR10 (Stolen)

MNIST (Benign)

Tiny-ImageNet (Benign)
Transfer Learning Attack

**Victim**

- Dataset
- DNN model

**Attacker**

- Attacker’s dataset
- Stolen model
- Attacker’s model

The attacker steals the model via transfer learning.
DEEP TASTER against Transfer Learning Attack

• DEEP TASTER is effective in identifying all transfer learning attack cases as the theft image rate is above 50%
DEEP TASTER VS. DEEP JUDGE [5]

• Compare with DEEP JUDGE, a state-of-the-art fingerprinting scheme
  - With 8 attack cases and 5 benign cases
  - Report the number of successfully detected models out of 10 suspect models for each attack scenario
## DeepTaster vs. DeepJudge

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Suspect</th>
<th>DeepTaster (Ours)</th>
<th>DeepJudge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benign</strong></td>
<td>MNIST</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST SAA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST MFA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST MPA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Tiny ImageNet</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td><strong>Stolen</strong></td>
<td>CIFAR10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 DAA</td>
<td>9</td>
<td>FAIL (4)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 SAA</td>
<td>9</td>
<td>FAIL (1)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 TLA</td>
<td>10</td>
<td>FAIL (0)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 MFA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 MPA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 DATLA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 TLPA</td>
<td>10</td>
<td>FAIL (0)</td>
</tr>
</tbody>
</table>
**DEEP TASTER vs. DEEP JUDGE**

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Suspect</th>
<th>DEEP TASTER (Ours)</th>
<th>DEEP JUDGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>MNIST</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST SAA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST MFA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MNIST MPA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Tiny ImageNet</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>CIFAR10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 SAA</td>
<td>9</td>
<td>FAIL (4)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 TLA</td>
<td>10</td>
<td>FAIL (1)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 MFA</td>
<td>10</td>
<td>FAIL (0)</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 MPA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 DATLA</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 TLPA</td>
<td>10</td>
<td>FAIL (0)</td>
</tr>
</tbody>
</table>

**DEEP TASTER** is effective in detecting eight attack scenarios, while **DEEP JUDGE** fails to detect four attack scenarios including transfer learning.
Limitations: Unseen Architecture

- **DEEPASTER** is not effective in detecting models trained using completely new or unseen architectures.
- To address this issue, we can consider more diverse and additional models for training our classifier.
Limitations: Adversarial Training

- **DEEPTASTER** is less robust against adversarial training.
Conclusion

Summary

• Propose a DNN fingerprinting method named **DEEP TASTER**

• Show the robustness of **DEEP TASTER** against eight attack scenarios

Evaluation

• **DEEP TASTER** shows resilience against eight attack scenarios

• **DEEP TASTER** considerably outperforms **DEEP JUDGE** in most scenarios

**DEEP TASTER**

• **DEEP TASTER** is a DNN fingerprinting method designed to identify known model architectures trained on stolen datasets

• **DEEP TASTER** generates adversarial images, transforms them into the DFT domain, and uses these transformed images to discern the unique characteristics of the dataset used to train a suspect model

Github codes are available on the following QR code

https://github.com/qkrtjsgp08/DeepTaster

Thanks!

Q&A