DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories

Adnan Siraj Rakin*\(^1\), Md Hafizul Islam Chowdhuryy\(^*2\), Fan Yao\(^2\), Deliang Fan\(^1\)

*Joint First Authors

\(^1\)Arizona State University (asrakin@asu.edu, dfan@asu.edu)
Website: https://dfan.engineering.asu.edu/

\(^2\)University of Central Florida (reyad@knights.ucf.edu, Fan.Yao@ucf.edu)
Website: https://casrl.ece.ucf.edu
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup
• Results & Conclusion
Machine Learning (ML) Applications

- Machine Learning Applications:
  - Robotics
  - Medical Applications
  - Self-Driving Cars

- Machine Learning Cloud Services:
  - Amazon AWS AI
  - Google AI
  - Microsoft Azure ML
Adversarial Threats in ML:

### Model Tampering (Security)
- **External Threat (Input perturbation)**
  - Adversarial Examples (Madry et. al. ICLR-18)
- **Internal Threat (Weight perturbation)**
  - DeepHammer (Yao et. al. USENIX SEC-20)
- **Both (Trojan/Backdoor Attack)**
  - Trojan NN (Liu et. al. NDSS-18)

### Model Leakage (Privacy)
- **Model Inversion Attack**
  - Recover Data (Fredrikson et. al. CCS-15)
- **Membership Inference Attack**
  - Leak Training Data (Shokri et. al. S&P-17)
- **Model Extraction Attack**
  - Recover Model Architecture/Weights
    - DeepSniffer (Hu et. al. ASPLOS-20)

Model Extraction Attack Objective:

1. Create a substitute model to *mimic the functionality* of the target model with *limited dataset* (less than 10%).

2. The substitute model should have a *high accuracy and fidelity*.

3. The substitute model can generate *strong transferable adversarial examples* to attack the target model.
Remote Side channel Attack on ML Model

**Primary Goal of Prior Works:**
Recover model architecture (i.e., no. of layers/connections)

**Example:**
*Cache telepathy* [USENIX Security’20], *DeepSniffer* [ASPLOS’20]

**Opportunities:**
1. None of the existing remote side-channel works have successfully recovered fine-grained weight information.

2. Exfiltration of weight information can potentially be even more dangerous than leakage of architecture information.
Can we recover fine-grained weight information through the remote side channels?

How to utilize partial weight information to perform advanced model extraction?
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup
• Results & Conclusion
Threat Model

- Attacker *knows* the DNN model architecture.
- Attacker *does not know* gradient or model parameter information.
- Attacker *cannot query* the target model to get output scores.
- Attacker can run *userspace process* on the victim machine.
- System software are *benign and properly protected.*
DeepSteal Overview

**Attacker’s Knowledge Of $W_1$**

- **Initial stage**
  - Attacker’s initial knowledge is represented by question marks.
- **HammerLeak Bit Stealing**
  - After the initial stage, the attacker can retrieve a partial bit of the model.
- **Mean Clustering**
  - Further processing allows the attacker to obtain a more complete bit of the model.

**Victim’s target model**

1. **Victim Model**
2. **Hammer Leak**
3. **Mean Clustering Training**
4. **Substitute Model**
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup
• Results & Conclusion
Data Leakage through RowHammer

RowHammer-based fault injection

Bitflips are *data dependent*
Data Leakage through RowHammer

RowHammer-based fault injection

Vulnerable cell

Bitflip

Target (Victim)

Aggressor (Attacker)

Aggressor (Attacker)

DRAM Bank

Bitflips are data dependent
Data Leakage through RowHammer

RowHammer-based information leakage ($S \rightarrow \text{Secret}$)

Aggressor bit can be leaked based on the existence of bitflip (*RAMBleed, S&P’20*)
Challenges:

C1: RowHammer information leakage from generic victim application.

C2: Bulk data stealing from victim with large-scale memory footprint.
Generic RowHammering For Bit Leakage

1. Bit flip = $S \rightarrow 1$

2. No Bit flip = $S \rightarrow 0$
Generic RowHammering For Bit Leakage

1. Bit flip = \( S \rightarrow 1 \)

2. No Bit flip = \( S \rightarrow 0 \)

3. Bit flip = \( S \rightarrow 1 \)

4. No Bit flip = \( S \rightarrow 0 \)
HammerLeak Framework: Detailed

Anonymous Page Swapping

DRAM Bank

Swap space

Free Page

Victim Page

Attacker Page

Vulnerable cell
HammerLeak Framework: Detailed

Anonymous Page Swapping

DRAM Bank

- Free Page
- Victim Page
- Attacker Page
- Vulnerable cell

Swap space

P1
P2
P3
HammerLeak Framework: Detailed

Anonymous Page Swapping

Bitflip-aware Page Release

Pageset (LIFO)

DRAM Bank

Free Page  Victim Page  Attacker Page  Vulnerable cell
HammerLeak Framework: Detailed

Anonymous Page Swapping → Bitflip-aware Page Release → Deterministic Victim Relocation

Pageset (LIFO) → DRAM Bank → Swap space

- Free Page
- Victim Page
- Attacker Page
- Vulnerable cell
HammerLeak Framework: Detailed

Anonymous Page Swapping → Bitflip-aware Page Release → Deterministic Victim Relocation → Rowhammer-based Bit Recovery

DRAM Bank

- Free Page
- Victim Page
- Attacker Page
- Vulnerable cell
HammerLeak: Batched Page Release

Use smaller batch size: macro-anchor to further divide victim execution

Bulk data-stealing from application with large memory-footprint
HammerLeak: Leaking PyTorch Model Weights

Victim execution:

Attacker: Populate pageset

DNN Model
Conv → Linear → Output

Convolutional Layer

Linear Layer

Fbgemm Macro Kernel

qconv.cpp: PackedConvWeight
qlinear.cpp: PackedLinearWeight
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup
• Results & Conclusion
Why do we still need training?

- Problem: How to use the partial bit information recovered from HammerLeak?

- Solution: We propose a training algorithm to successfully utilize the stolen partial bit information.
Substitute Model Training:

1. Each weight has a projected range.
Substitute Model Training:

1. Each weight has a projected range.
Substitute Model Training:

1. Each weight has a projected range.
Substitute Model Training:

1. Each weight has a projected range.

2. Mean clustering penalty ensures the weights stay well within the projected range during training.

\[
\min_{\{W_l\}_{l=1}^L} \mathbb{E}_x \mathcal{L}(f(x; \{W_l\}_{l=1}^L), y) + \lambda \cdot \sum_{l=1}^L (||W_l - W_{l,\text{mean}}||)
\]

- loss penalty for Mean Clustering
Algorithm: Mean Clustering Training

- Weight Set-1: **All 8-bits recovered**
  No Training i.e., set the gradient of the weights to zero.

- Weight Set-2: **Partial bits recovered starting from most significant bits**
  Apply mean clustering penalty only for these set of weights.

- Weight Set-3: **No bit recovered or bit recovered without MSBs**
  Train w/o any clustering penalty.
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup
• Results & Conclusion
Experimental Setup

• Dataset: Popular vision datasets (e.g., *CIFAR-10/100, GTSRB*).

• Architecture: *ResNets and VGG*.

• Attacker Data: 8% training data available to train the substitute model.

• Training Platform: *PyTorch* running on *GeForce GTX 1080 Ti GPU* platform.

• Attack Platform: *Intel Haswell* series processor.

• Memory configuration: *Dual-channel DDR3*. 
**Evaluation Metrics:**

*Accuracy (%)*: Accuracy of the substitute model on test dataset.

*Fidelity (%)*: Percentage of test samples both the target and substitute model agree on their classification result.

*Adversarial Example Attack (%)*: Test accuracy of a target model on the adversarial test samples generated using the recovered substitute model as shown in the left figure.
Outline

• Background
• Threat Model and Overview
• System-level Attack
• Substitute Model Training
• Experimental Setup

• Results & Conclusion
Results: HammerLeak

HammerLeak Analysis:

- Bit leakage accuracy: 95.73% (Standard deviation: 0.74%).
- ResNet-18 weight leakage rate.

**Figure:** Distribution of weights with MSB recovered across 21-layers
Results: Mean Clustering Training

- Increasing attack round generates effective substitute model with higher accuracy & fidelity.
- At 4000 rounds, we could achieve similar adversarial example attack performance as the white-box attack.

<table>
<thead>
<tr>
<th>CIFAR-10 (ResNet-18)</th>
<th>Time (Days)</th>
<th>Recovered (MSB) (%)</th>
<th>Accuracy (%)</th>
<th>Fidelity (%)</th>
<th>Adversarial Example Attack (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Only</td>
<td>-</td>
<td>0</td>
<td>73.18</td>
<td>74.29</td>
<td>61.33</td>
</tr>
<tr>
<td>1500 Rounds</td>
<td>3.9</td>
<td>60</td>
<td>76.61</td>
<td>77.56</td>
<td>50.4</td>
</tr>
<tr>
<td>3000 Rounds</td>
<td>7.8</td>
<td>80</td>
<td>86.93</td>
<td>88.51</td>
<td>8.13</td>
</tr>
<tr>
<td>4000 Rounds</td>
<td><strong>10.4</strong></td>
<td><strong>90</strong></td>
<td><strong>89.59</strong></td>
<td><strong>91.6</strong></td>
<td><strong>1.61</strong></td>
</tr>
<tr>
<td>Best-Case (White Box)</td>
<td>-</td>
<td>100</td>
<td>93.16</td>
<td>100.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Increasing attack round generates effective substitute model with higher accuracy & fidelity. At 4000 rounds, we could achieve similar adversarial example attack performance as the white-box attack.
# Comparison with Existing Methods:

<table>
<thead>
<tr>
<th>Recovery Method</th>
<th>Accuracy (%)</th>
<th>Adversarial Example Attack (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture only e.g., DeepSniffer (ASPLOS 20)</td>
<td>72.68</td>
<td>62.68</td>
</tr>
<tr>
<td><em>DeepSteal (Architecture + Partial Weight-Bit Information)</em></td>
<td>90.35</td>
<td>1.2</td>
</tr>
</tbody>
</table>

- DeepSteal shows ~18% improvement in accuracy compared to the existing remote side-channel attacks which only focus on recovering the architecture only information of DNN.

- Fine-grained bit information significantly improves the adversarial attack performance as well.
Comparison with Existing Methods:

<table>
<thead>
<tr>
<th>Attack Threat Model</th>
<th>Adversarial Example Attack (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Box (Transfer Cui et. al.)</td>
<td>20.47</td>
</tr>
<tr>
<td>White-Box (PGD Madry et. al.)</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>DeepSteal (ours)</strong></td>
<td><strong>1.2</strong></td>
</tr>
</tbody>
</table>

- DeepSteal threat model falls in the gray-box zone (architecture known) between white-box and black-box attack.
- Fine-grained bit information achieves almost similar success rate as the white-box attack.
Conclusion:

• DeepSteal with the exploitation of a remote side channel, for the first time, can exfiltrate fine-grained weight information in bulk from DNN model.

• DeepSteal can recover substitute model with high accuracy and fidelity (~90%).

• The adversarial examples generated from the substitute model is as effective as a white-box attack.

• Our proposed attack opens a practical solution to identical model recovery and urges the community to invest in future defense solutions.
Thank You & Questions?

Adnan Siraj Rakin
Email: asrakin@asu.edu

Md Hafizul Islam Chowdhuryy
Email: reyad@knights.ucf.edu

Fan Yao, Ph.D.
Email: Fan.Yao@ucf.edu
Website: https://casrl.ece.ucf.edu

Deliang Fan, Ph.D.
Email: dfan@asu.edu
Website: https://dfan.engineering.asu.edu/

This work is supported in part by the National Science Foundation under Grant No. 2019548, No. 2019536, No. 1931871, and No. 2144751.