Hermes Attack: Steal DNN Models with Lossless Inference Accuracy

Yuankun Zhu#, Yueqiang Cheng*, Husheng Zhou§, Yantao Lu+
#University of Texas at Dallas, *Baidu Security, §Vmware, +Syracuse University
Motivations

Autonomous Driving

Privatization Deployments
Motivations

$4.5/hr per TPU

$400,000

DNN Model
Motivations

• All existing attacks can only reconstruct partial model
• Hermes Attack is the first work that can fully steal DNN model with lossless interface accuracy
Challenges and Solutions

**C1**: Closed-source Code and Undocumented Data Structures

**S1**: Reverse Engineering

**C2**: Numerous Noises and Chaotic Orders in PCIe packets

**S2**: Sanitization and Order Correction

**C3**: Semantic Loss in PCIe Traffic

**S3**: Semantic Reconstruction
How GPU Works

O1: **Command driven** interaction working mode
O2: All data and code are passed through **PCIe bus**
**PCIe Packet Relationship**

---

**Example of Read Request TLP**

| 3 1 0 9 | 2 8 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 9 | 1 8 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0        | For| mat|     | 0  | TC | 0  | T | E | Attr | 0  | Length |
| Requester ID |     |     |     |    |    |    |   |   |      |    |         |
| Address[32:23] |     |     |     |    |    |    |   |   |      |    |         |
| Address[31:2] |     |     |     |    |    |    |   |   |      |    |         |

---

**Example of Completion TLP**

| 3 1 0 9 | 2 8 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 9 | 1 8 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0        | For| mat|     | 0  | TC | 0  | T | E | Attr | 0  | Length |
| Completer ID |     |     |     |    |    |    |   |   |      |    |         |
| Requester ID |     |     |     |    |    |    |   |   |      |    |         |

**Address and Payload are connected through the same tag value**
Terminologies

PCle Packet

<table>
<thead>
<tr>
<th>PCle Header</th>
<th>PCle Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCle Header</td>
<td>PCle Payload</td>
</tr>
<tr>
<td>PCle Header</td>
<td>PCle Payload</td>
</tr>
<tr>
<td>PCle Header</td>
<td>PCle Payload</td>
</tr>
</tbody>
</table>

Command Header

62200220 ...

Command Data

xxxxxxxx ...

xxxxxxxx ...

xxxxxxxx ...

xxxxxxxx ...

<table>
<thead>
<tr>
<th>For mat</th>
<th>Type</th>
<th>TC</th>
<th>T</th>
<th>E</th>
<th>Attr</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>T</td>
<td>E</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completer ID</th>
<th>Status</th>
<th>Byte Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requester ID</th>
<th>Tag</th>
<th>Last BE</th>
<th>First BE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reserved
Attack Overview
Attack Overview
PCIe Bus Snooping Device: PCIe protocol Analyzer

Out of order!
Sort Payload

Sort by the upstream packet ID

Use Packet ID locate corresponding data packet

Use Tag locate corresponding downstream packet

**Packet ID**

<table>
<thead>
<tr>
<th>Data Packets</th>
<th>Completion Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>P2</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>103</td>
<td>103</td>
</tr>
</tbody>
</table>

- 100: 0xXXXX Xxxxxxxxxxx 79
- 101: 0xXXXX Xxxxxxxxxxx 80
- 102: 0xXXXX Xxxxxxxxxxx 10
- 103: 0xXXXX Xxxxxxxxxxx 10

**Packet ID**

<table>
<thead>
<tr>
<th>Request Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3'</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

- 100: 0xXXXX Xxxxxxxxxxx 80
- 101: 0xXXXX Xxxxxxxxxxx 79
- 102: 0xXXXX Xxxxxxxxxxx 65
- 103: 0xXXXX Xxxxxxxxxxx 110
Attack Overview
Numerous Noises

- In a Traffic sample:
  - Number of PCIe packets: 1,077,757
  - Number of useful packets: ~20,000
  - 98.4% of the packets are noise

Focus on specific GPU Commands!
GPU is controlled by CPU using commands. Command types includes:

- GPU Initialization Command
- Synchronization Command
- Data Movement Command
- Kernel Launch Command

We only need Data Movement Commands, noted as D commands, and Kernel Launch Commands, noted as K commands.

Identify headers of D cmds and K cmds through offline training.
Example of a GPU command header

<table>
<thead>
<tr>
<th>Address</th>
<th>Payloads</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000008:292B7F00</td>
<td>62200220 0C7F0000 50182EBA 50200220 20010000 01000000 6C200120 41000000 6D200860</td>
<td>Maximum 4096 Bytes</td>
</tr>
<tr>
<td>00000008:292B7F80</td>
<td>...XXXXXXXX XXXXXXXXX XXXXXXXXX XXXXXXXXX XXXXXXXXX XXXXXXXXX XXXXXXXXX XXXXXXXXX ...</td>
<td></td>
</tr>
</tbody>
</table>

: Command Header
Command Extraction

**Command Header Noise:** when a command header has been split into two packets, the noise may appear between these two packets.

**Command Data Noise:** noise packet appeared among command data packets.

Check address consistency!
Attack Overview

1. PCIe Interceptor
   - Raw Traffic
   - Sorted Traffic

2. Traffic Processing
   - Header Extraction
   - Command Extraction
     - Traffic
     - K Commands
     - D Commands

3. Reconstruction
   - Semantic Reconstruction
   - Model Reconstruction
     - Architecture
     - Hyper-Parameters
     - Parameters

Generated Model
Model Reconstruction

Step 1: Architecture

Step 2: Hyper-params

Step 3: Parameters
Semantic Reconstruction Challenges

❖ Layer type information is lost in PCIe traffic
❖ Every layer is completed through several kernels.
❖ Every Kernel is launched through a *K command*.
❖ Every *K* Command include an address pointed to its kernel binaries.
Offline Database Generation

❖ Determine kernel types of kernel binaries
  ➢ Extract corresponding kernel binaries from PCIe traffic.
  ➢ Obtain kernel traces from profiling tools.

❖ Determine kernels of layers
  ➢ Repeat inference procedure on different models.
  ➢ Obtain kernel trace of each layer.

❖ Put <kernel binaries, layer types> pairs into database.
Extract Architecture

• Build Data Flow Graph
  • Treat kernel as vertice
  • Treat Input address and output address as flow-from and flow-to
• Substitute kernel vertice as DNN layers

- conv
- Bias
- conv
- ReLU
- Pool
- ReLU
- Bias
- Add
• Build Data Flow Graph
  • Treat kernel as vertex
  • Treat Input address and output address as flow-from and flow-to
• Substitute kernel vertex as DNN layers
Model Reconstruction

Step 1: Architecture

Step 2: Hyper-params

Step 3: Parameters
Extract Hyper-Parameters

❖ Directly obtained from PCIe
  ➢ Fetch from command data, use identified offset.
  ➢ eg. Kernel size, Strides, Filters, etc.

❖ Infer
  ➢ Check if related kernel exist
  ➢ eg. Activation function, Use_bias
Model Reconstruction

Step 1: Architecture
Step 2: Hyper-params
Step 3: Parameters
• **Observation:**
  • Weights and bias are not called by kernels directly, they are first called by `CudaMemcpyDeviceToDevice`.
  • This API is also used for **kernel launch command** to complete.

*Model Reconstruction ✔*

Evaluation

- Operating System: Ubuntu 16.04
- Tested GPU platform:
  - NVIDIA GeForce GT 730
  - NVIDIA GeForce GTX 1080 Ti
  - NVIDIA GeForce RTX 2080 Ti
- Tested DNN models:
  - MNIST
  - VGG-16
  - ResNet-20
- GPU Programming Interface: CUDA 10.1
- GPU Driver: NVIDIA 418.43
- DNN platform: keras + tensorflow
Evaluation

VGG Architecture Comparison
Evaluation

ResNet Architecture Comparison
## Evaluation

### Identity Evaluation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Model</th>
<th>Original</th>
<th>Reconstructed</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>GT 730</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1080 Ti</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>2080 Ti</td>
</tr>
<tr>
<td>MNIST Accuracy</td>
<td></td>
<td>98.25%</td>
<td>Same Accuracy</td>
</tr>
<tr>
<td>VGG</td>
<td></td>
<td>93.59%</td>
<td>Same Accuracy</td>
</tr>
<tr>
<td>ResNet</td>
<td></td>
<td>91.45%</td>
<td>Same Accuracy</td>
</tr>
<tr>
<td>Inference Times(s)</td>
<td></td>
<td>2</td>
<td>Same Efficiency</td>
</tr>
<tr>
<td>MNIST</td>
<td></td>
<td>2</td>
<td>65</td>
</tr>
<tr>
<td>VGG</td>
<td></td>
<td>2</td>
<td>63</td>
</tr>
<tr>
<td>ResNet</td>
<td></td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>21</td>
</tr>
</tbody>
</table>
# Evaluation

## Performance Evaluation

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>VGG</th>
<th>ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Platform</strong></td>
<td>GT 730</td>
<td>1080 Ti</td>
<td>2080 Ti</td>
</tr>
<tr>
<td><strong># of D Commands</strong></td>
<td>25,680</td>
<td>28,590</td>
<td>24,342</td>
</tr>
<tr>
<td><strong># of K Commands</strong></td>
<td>216</td>
<td>139</td>
<td>181</td>
</tr>
<tr>
<td><strong># of Completion Packets</strong></td>
<td>1,077,756</td>
<td>2,244,115</td>
<td>2,959,613</td>
</tr>
<tr>
<td><strong>Generation Time (min)</strong></td>
<td>5</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

- **GT 730**: GPU model
- **1080 Ti**: GPU model
- **2080 Ti**: GPU model