Securing Architecture Supported ML-systems

Nael Abu-Ghazaleh
Some Trends (Arch2030)

Acceptance of Moore’s law ending

Machine learning critical workload

Specialization happening at scale

Cloud is ubiquitous

3D devices, quantum, and DNA computing
What is at the intersection of Security, ML and Architecture?

Intersection of ML/AI, Arch and Security

1. AI for Architecture
2. Architecture for AI

AIDArc

Machine Learning

Security

Architecture

??
Architecture and ML

• Architecture for ML
  – Training and inference are expensive
    • GPUs, accelerators, ...
  – Real-time response, low-power devices, ...

• ML for Architecture
  – Use ML for design, resource allocation, prediction, security, reconfiguration, ...
ML and Security: Trustworthy AI

• Security of ML: ML is deployed everywhere
  – how can we ensure security and privacy?

• What are the threat models facing AI?
  – How can we mitigate them?

• ML for security is of course also a thing
Adversarial attacks

“Yes”

“Yes!!”

“Yes”
Membership Inference attacks

Private data

Model

Was this specific picture in the training set?
Model Extraction attacks

Data Owner → Train → ML Service

Data → Model

\[ f(X_t) \]

\[ f(X_i) \]

\[ f(X_z) \]
Architecture & Security

Vulnerabilities originating in architecture

- Side and covert channels
- Speculation attacks
- Fault injection
- Hardware trojans, ...

Defenses rooted in architecture

- Do no harm
  - Avoid vulnerabilities in architecture/HW
- Help software
  - Security abstractions/mechanisms
  - Computational support for expensive defenses

Attacks

Defenses

9
Three examples

1. Attacking ML used in computer architecture
2. Microarchitectural attacks on ML workloads
3. Approximate computing to harden ML workloads
Example 1: Evasion Resilient Hardware Malware Detectors
Malware is Everywhere!

Over 350,000 malware registered every day!

Hardware Malware Detectors (HMDs)

Use **Machine Learning**: detect malware as computational anomaly

Use low-level features collected from the hardware

Can be always-on without adding performance overhead

Many research papers including ISCA, HPCA, MICRO, DAC, ICCAD
Attacking HMDs

Can malware evade detection?

Reverse-engineer HMDs

Develop evasive malware
REVERSE ENGINEERING
How to Reverse Engineer HMDs?

Challenges:
- We don’t know the detection period
- We don’t know the features used
- We don’t know the detection algorithm

Approach:
1. Train different classifiers
2. Derive specific parameters as an optimization problem
Current generation of HMDs can be reverse engineered
EVADING HMDS
How to Create Evasive Malware?

• Challenges:
  - We don’t have malware source code
  - We can’t decompile malware because its obfuscated

• Our approach:
What we Should Add to Evade?

Current generation of HMDs is vulnerable to evasion attacks!
Overview (cont’d)

Can malware evade detection?
- Reverse-engineer HMDs
- Develop evasive malware

Can we make HMDs robust to evasion?
- Yes! using RHMD
  1- Provably harder to reverse-engineer
  2- Robust to evasion
Overview of RHMDs

RHMD

HMD 1
HMD 2
...
HMD n

Pool of diverse HMDs
Overview of RHMDs

- Detection period
- Features vector
- RHMD
- Number of committed instructions

Features vector: [ ] [ ] [ ] ...

Selector

Input → RHMD

Output

RHMD: HMD 1, HMD 2, ..., HMD n
Overview of RHMDs

Detection period

Features vector

Number of committed instructions

RHMD

Input

Output

Selector
Overview of RHMDs

Detection period

Features vector

0

Number of committed instructions

Features vector

Selector

Input

Output

RHMD

HMD 1

HMD 2

\ldots

HMD n
Overview of RHMDs

Diversify by Different:

1- Features
2- Detection periods
RHMD is Resilient to Evasion

No attack

Accuracy

Attack
1 instruction at each basic block

Intensive attack
10 instructions at each basic block
Lessons/Future directions

• ML used in architectures and systems vulnerable
  – Imagine other uses of ML

• ...but it's possible to design robust classifiers
Example 2: Model extraction via Side channel attacks
Overview

• Can side channel attacks operate on new computational environments?
  – GPUs, TPUs, NPUs, heterogeneous SoCs

• Can side channel attacks compromise ML workloads?
### Threat Model [CCS 2018]

<table>
<thead>
<tr>
<th>Programming Interfaces</th>
<th>Example Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA/Graphics Spy</td>
<td>GPU rendering</td>
</tr>
<tr>
<td>on Graphics Victim</td>
<td>(Web Browsers, …)</td>
</tr>
<tr>
<td></td>
<td>GPU Accelerated</td>
</tr>
<tr>
<td></td>
<td>Computations</td>
</tr>
<tr>
<td></td>
<td>(DNN, Encryption, …)</td>
</tr>
</tbody>
</table>
Leakage Vectors

• **Memory allocation API:**
  Exposes the amount of available physical memory on the GPU

• **Timing:**
  Recording the time of memory allocation events

• **GPU hardware performance counters:**
  Memory, instruction, multiprocessor, cache and texture metrics.
Reverse Engineering Co-location

- Reverse engineering the co-location of two concurrent applications:

  **Graphics App1**
  - CPU Code
  - GPU Code (vertex and fragment shaders)

  **Graphics App2**
  - CPU Code
  - GPU Code (vertex and fragment shaders)

  **CPU Code:**
  Read the pixel colors and decode the information

  **GPU Code (fragment shader):**
  Use OpenGL extensions to read ThreadID, WarpID, SMID and time of each fragment (pixel/thread) and encode this information in the color of each pixel.

  ```
  glReadPixels(...);
  SMID = float(gl_SMIDNV)/3.0;
  clock = float(clock2x32ARB().y)/4294967295.0;
  ThreadID = float(gl_ThreadInWarpNV)/32.0;
  color = vec4(0, ThreadID, SMID, clock);
  ```

Two graphics applications whose workloads do not exceed the GPU hardware resources can co-locate concurrently.
Attack 1: Website Fingerprinting

• Browsers use GPU to accelerate rendering.

• A content-related pattern of memory allocations on the GPU.
GPU Memory Allocation Trace
## Classification

- The classification results for Memory API based website fingerprinting attack on 200 Alexa Top Websites:
  - Gaussian Naive Bayes (NB)
  - K-Nearest Neighbor with 3 neighbors (KNN-3)
  - Random Forest with 100 estimators (RF)

<table>
<thead>
<tr>
<th></th>
<th>FM %</th>
<th>Prec %</th>
<th>Rec %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>µ (σ)</td>
<td>µ (σ)</td>
<td>µ (σ)</td>
</tr>
<tr>
<td>NB</td>
<td>83.1 (13.5)</td>
<td>86.7 (20.0)</td>
<td>81.4 (13.5)</td>
</tr>
<tr>
<td>KNN-3</td>
<td>84.6 (14.6)</td>
<td>85.7 (15.7)</td>
<td>84.6 (14.6)</td>
</tr>
<tr>
<td>RF</td>
<td>89.9 (11.1)</td>
<td>90.4 (11.4)</td>
<td>90.0 (12.5)</td>
</tr>
</tbody>
</table>
CUDA-CUDA Attack Overview
Neural Network Model Extraction

**Leakage:** GPU performance counters

**Victim:** A CUDA-implemented back-propagation (Rodinia benchmark)

**Spy:** Launches several hundred consecutive CUDA kernels

**Methodology:**
- **Co-locate:** Reverse engineer GPU hardware schedulers to co-locate on each SM
- **Create contention:** Different threads (or warps) utilize different hardware resources in parallel to create contention
- **Measure:** Collecting one vector of performance counter values from each spy kernel
The classification results for identifying the number of neurons through the side channel attack:

<table>
<thead>
<tr>
<th></th>
<th>FM %</th>
<th>Prec %</th>
<th>Rec %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>µ (σ)</td>
<td>µ (σ)</td>
<td>µ (σ)</td>
</tr>
<tr>
<td>NB</td>
<td>80.0 (18.5)</td>
<td>81.0 (16.1)</td>
<td>80.0 (21.6)</td>
</tr>
<tr>
<td>KNN-3</td>
<td>86.6 (6.6)</td>
<td>88.6 (13.1)</td>
<td>86.3 (7.8)</td>
</tr>
<tr>
<td>RF</td>
<td>85.5 (9.2)</td>
<td>87.3 (16.3)</td>
<td>85.0 (5.3)</td>
</tr>
</tbody>
</table>

Input layer size varying in the range between 64 and 65536 neurons collecting 10 samples for each input size
Lessons/Future Directions

• Microarchitectural attacks can compromise ML workloads

• Other attacks possible (e.g., Rowhammer)

• New architectures can expose attack surfaces
Direction 3: Approximate computing for ML Robustness
Defenses against adversarial attacks

Robust Models

Data Filtering and Transformation
Defenses against adversarial attacks

- **Adversarial training**
  - Retrain the model with constructed adversarial examples
  - Refine accuracy on adversarial examples
  - Might also refine clean accuracy

- **Gradient masking-based**
  - Regularize or smooth labels to make less sensitive models
  - Distillation
  - Contrastive network

- **Run time defenses**
  - Fails to defend but makes white box attacks harder

- Can be computationally expensive

- Attacks harder
Approximate computing to the rescue

(a) 

\[ A \quad B \quad nCi \quad t \quad u \quad n \quad t \quad Co \quad Sum \]

(b) 

\[ A \quad B \quad Ci \quad Sum \quad Cou \]

Input

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>0</th>
<th>4</th>
<th>3</th>
<th>6</th>
<th>2</th>
<th>7</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>0</td>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td></td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kernel

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Output

\[(1\times2) + (0\times4) + (3\times0) + (2\times1) + (1\times2) + (1\times1) + (5\times0) + (0\times3) + (4\times0) = 7\]
Approximation error

<table>
<thead>
<tr>
<th>Multiplier</th>
<th>LeNet5</th>
<th>AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact multiplier</td>
<td>97.78%</td>
<td>81%</td>
</tr>
<tr>
<td>Ax-FPM</td>
<td>97.77%</td>
<td>80%</td>
</tr>
</tbody>
</table>
Grey box attack
# Grey box attack results

TABLE IV: Classification accuracy under various Grey-box attacks on the MNIST subset.

<table>
<thead>
<tr>
<th>Attack method</th>
<th>Exact LeNet-5</th>
<th>Approximate LeNet-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;W</td>
<td>0%</td>
<td>99%</td>
</tr>
<tr>
<td>PGD</td>
<td>0%</td>
<td>72%</td>
</tr>
<tr>
<td>HSJ</td>
<td>0%</td>
<td>98%</td>
</tr>
<tr>
<td>FGSM</td>
<td>0%</td>
<td>88%</td>
</tr>
<tr>
<td>JSMA</td>
<td>0%</td>
<td>91%</td>
</tr>
<tr>
<td>LSA</td>
<td>0%</td>
<td>82%</td>
</tr>
<tr>
<td>BA</td>
<td>0%</td>
<td>83%</td>
</tr>
</tbody>
</table>

TABLE V: Classification accuracy under various Grey-box attacks on the CIFAR-10 subset.

<table>
<thead>
<tr>
<th>Attack method</th>
<th>Exact AlexNet</th>
<th>Approximate AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;W</td>
<td>0%</td>
<td>87%</td>
</tr>
<tr>
<td>PGD</td>
<td>0%</td>
<td>69%</td>
</tr>
<tr>
<td>HSJ</td>
<td>0%</td>
<td>88%</td>
</tr>
<tr>
<td>FGSM</td>
<td>0%</td>
<td>62%</td>
</tr>
<tr>
<td>JSMA</td>
<td>0%</td>
<td>68%</td>
</tr>
<tr>
<td>LSA</td>
<td>0%</td>
<td>64%</td>
</tr>
<tr>
<td>BA</td>
<td>0%</td>
<td>63%</td>
</tr>
</tbody>
</table>
Confidence distribution
Whitebox attack

Approximate classifier architecture and parameters

Generating the adversarial example

Clean input correctly classified by the exact classifier

Approximate implementation

Adversarial example misclassified by the approximate classifier

Attack on the approximate classifier
Approximate model

Exact model

MSE

PSNR (dB)

1 2 3 4 5 6 7 8 9 10

Images

1 2 3 4 5 6 7 8 9 10

Images
Approximate computing is more efficient
Conclusions

• Interesting research at the intersection of ML, Architecture and Security

• Three examples:
  – Attacking (and protecting) ML used in computing systems
  – Microarchitectural model extraction attacks
  – More efficient implementation of defenses

• There are many other research cross-pollination possibilities among these three areas
Collaborators/Students

Khaled Khasawneh
Shirin HajiAmin Shirazi
Hoda Naghibijouybari
Amira Gusemi
Ihsen Alouini
Dmitry Ponomarev
Lei Yu
Zhiyun Qian
THANKS – QUESTIONS?