Reverse Engineering AI Models

Kang Li
kangli.ctf@gmail.com

Collaborators: Deyue Zhang, Jiayu Qian, Yufei Chen
About Me

Professor of Computer Science at UGA

Founder of the SecDawgs, Disekt CTF Teams

Founding Mentor of xCTF and Blue-Lotus

2016 DARPA Cyber Grand Challenge Finalist
Deep Learning AI Applications
AI Application Security
AI Application Threat Landscape (Kang’s view)

- AI App Security Risk
  - Model Security
    - Adversarial ML
    - Model Backdoor
    - Model Theft
  - Implementation Security
    - Sensor Security
    - Flaws in Framework
    - Logical Flaws
  - Data Integrity Security
    - Data Poisoning
    - Scaling Attack
    - Risk over Network
My Prior Work

AI App Security Risk

Model Security
- Adversarial ML
- Model Backdoor
- Model Theft
- Sensor Security
- Flaws in Framework
- Logical Flaws

Data Integrity Security
- Data Poisoning
- Scaling Attack
- Risk over Network
Security Risks in Deep Learning Implementations

1ST DEEP LEARNING AND SECURITY WORKSHOP

May 24, 2018

co-located with the
39th IEEE Symposium on Security and Privacy

Sample Deep Learning Application (CAFFE ImageNet)

Output:

---------- Prediction for examples/images/cat.jpg ----------
0.3134 - "n02123045 tabby, tabby cat"
0.2380 - "n02123159 tiger cat"
0.1235 - "n02124075 Egyptian cat"
0.1003 - "n02119022 red fox, Vulpes vulpes"
0.0715 - "n02127052 lynx, catamount"

https://github.com/BVLC/caffe/tree/master/examples/cpp_classification
# Threat Example 1 -- DoS attack
# bulldog_crash [cause classification binary to crash]
./classification.bin models/bvlc_reference_caffenet/deploy.prototxt
models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt
./poc_samples/bulldog_crash
---------- Prediction for ./poc_samples/bulldog_crash ----------
Segmentation fault (core dumped)

# Threat Example 2 -- Evasion attack
# bulldog_sh [cause classification to misclassify to an arbitrary category]
# [Here the classification produced a class "Flying Pig" made up by attackers]
./classification.bin models/bvlc_reference_caffenet/deploy.prototxt
models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt
./poc_samples/bulldog_fp
---------- Prediction for ./poc_samples/bulldog_fp ----------
0.98 - "n03770679 flyingpig"

# Threat Example 3 -- Exploitation Attack
# bulldog_sh [cause classification to generate a local shell]
./classification.bin models/bvlc_reference_caffenet/deploy.prototxt
models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt
./poc_samples/bulldog_sh
---------- Prediction for ./poc_samples/bulldog_sh ----------
$ uname -a
Linux ctf-box 4.4.0-31-generic #50~14.04.1-Ubuntu SMP
Wed Jul 13 01:07:32 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
$ exit
# Threat Example 1 -- DoS attack
# bulldog_crash [cause classification binary to crash]
./classification.bin models/bvlc_reference_caffenet/deploy.prototxt
models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt
./poc_samples/bulldog_crash
---------- Prediction for ./poc_samples/bulldog_crash ----------
Segmentation fault (core dumped)

# Threat Example 2 -- Evasion attack
# bulldog_sh [cause classification to misclassify to an arbitrary category.]
# [Here the classification produced a class "Flying Pig" made up by attackers]
./classification.bin models/bvlc_reference_caffenet/deploy.prototxt
models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt
./poc_samples/bulldog_fp
---------- Prediction for ./poc_samples/bulldog_fp ----------
0.98 - "n03770679 flyingpig"

$ uname -a
Linux ctf-box 4.4.0-31-generic #50~14.04.1-Ubuntu SMP
Wed Jul 13 01:07:32 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
$ exit
My Prior Work

AI App Security Risk

Model Security
- Adversarial ML
- Model Backdoor
- Model Theft
- Sensor Security
- Flaws in Framework
- Logical Flaws

Implementation Security
- Data Integrity Security
- Data Poisoning
- Scaling Attack
- Risk over Network
Security Risks in Data Pre-Processing (Scaling)

Data Scaling Attacks in Deep Learning Applications
Data Flow in Deep Learning Image Applications
Scaling Functions Provided by Frameworks

```python
def read_tensor_from_image_file(file_name, input_height=299, input_width=299,
                             input_mean=0, input_std=255):
    input_name = "file_reader"
    output_name = "normalized"
    file_reader = tf.read_file(file_name, input_name)
    if file_name.endswith(".png"):
        image_reader = tf.image.decode_png(file_reader, channels = 3,
                                           name='png_reader')
    elif file_name.endswith(".gif"):
        image_reader = tf.squeeze(tf.image.decode_gif(file_reader,
                                                      name='gif_reader'))
    elif file_name.endswith(".bmp"):
        image_reader = tf.image.decode_bmp(file_reader, name='bmp_reader')
    else:
        image_reader = tf.image.decode_jpeg(file_reader, channels = 3,
                                             name='jpeg_reader')
    float_caster = tf.cast(image_reader, tf.float32)
    dims_expander = tf.expand_dims(float_caster, 0);
    resized = tf.image.resize_bilinear(dims_expander, [input_height, input_width])
    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
    sess = tf.Session()
    result = sess.run(normalized)
    return result
```

TensorFlow Example

```
tf.image.resize_bilinear(dims_expander, [input_height, input_width])
```
Attack Sample (Data Poisoning)
Evading Attack Example

✓ Description: {
  "tags": [
  "animal", "mammal", "wolf", "looking"
  ],
  "captions": [
  {
    "text": "a close up of a wolf",
    "confidence": 0.707954049
  }
  ]
}

✓ Tags: [...,
  {
    "name": "wolf",
    "confidence": 0.981169641
  }
]
AI Model Security
AI & Models

MNIST


ImageNet

https://github.com/BVLC/caffe/tree/master/examples/cpp_classification

NVIDIA PX DAVE-2

Sample Neural Network (LeNet-5) Architecture

Sample Model Files

Model Layers
(lenet_deploy.prototxt)
Sample Model Files

```cpp
1 name: "LeNet"
2 layer {
3   name: "data"
4   type: "Input"
5   top: "data"
6   input_param { shape: { dim: 1 dim: 1 dim: 28 dim: 28 } }
7 }

8 layer {
9   name: "conv1"
10  type: "Convolution"
11  bottom: "data"
12  top: "conv1"
13  param {
14    lr_mult: 1
15  }
16  param {
17    lr_mult: 2
18  }
19  convolution_param {
20    num_output: 20
21    kernel_size: 5
22    stride: 1
23    weight_filler {
24      type: "xavier"
25    }
26    bias_filler {
27      type: "constant"
28    }
29  }
```
Sample Model Files

Model Parameters
(lenet.caffemodel)
Sample Model Files

Model Parameters (lenet.caffemodel)

Parameters translated to plain text format

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  blobs {
    data: 0.063789606
    data: 0.41717789
    data: 0.24037249
    ...
    data: 0.0044610952
    shape {
      dim: 10
      dim: 500
    }
  }
  blobs {
    data: -0.0088488162
    data: -0.0015859469
    data: -0.015499314
    ...
    data: -0.0088488162
    data: -0.0015859469
    data: -0.015499314
    ...
  }
```

AI Application Security Threats

AI App Security Risk

- Model Security
  - Adversarial ML
  - Model Backdoor
  - Model Theft

- Implementation Security
  - Sensor Security
  - Flaws in Framework
  - Logical Flaws

- Data Integrity Security
  - Data Poisoning
  - Scaling Attack
  - Risk over Network
Threat related to AI Models

- model inference attack
- model inverse attack
- membership inference attack
- Reverse engineering attack
- software vulnerability
Attacks Related to Deep Learning Models

- Valuable Assets of AI Applications
- Models and Parameters (and Training data)
- Academics have paid substantial attentions at the danger of models and parameters leaking.


https://github.com/ftramer/Steal-ML
Model Inference Attack

- Attack on cloud-based AI models
- Selectively construct queries to Infer algorithms and parameters
- Targeted Algorithms
  - Logistic Regressions, SVMs, Decision Trees


https://github.com/ftramer/Steal-ML
Model Inversion Attack

- Infer information related to training data through specially crafted queries


https://github.com/ftramer/Steal-ML
Example: Model Inversion Attack

- Goal: to infer information in training data
- Experiments for Model Inversion Attack
  - Photo Images from 40 people
  - Simple Neural Network (one hidden layer)

Other Academic Works

- **Inferring deep learning hyper-parameters**
  

- **Inferring membership info in machine learning training data**
  

Threat related to AI Models

- Model inference attack
- Model inverse attack
- Membership inference attack
- Reverse engineering attack
- Software vulnerability
Some APPs that use AI Models

- Caffe
- Caffe2
- TensorFlow Mobile
- TF Lite
- MegVii
- SenseTime

Mixed Platform
If online processing is turned off, images are processed using a computing power of your device only. This could lead to lowering of a processing speed and unavailability of some styles.
Local AI Models ?!
AI Model Files Found

- Locally stored models in APKs
  - Plain framework model files (*.caffemodel, *.pb)
  - Plain but “obfuscated” filenames (e.g. *.mp3)
  - Encrypted model files
    - often using AES, with hard-coded but hidden keys
  - Model files for specific AI hardware (*.dlc, *.cambricon, *.pie)

- On-demand, model files downloaded from network
Local Model Files

• Built-in Model Files

<table>
<thead>
<tr>
<th>APK</th>
<th>com.----.camera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AndroidManifest.xml</td>
</tr>
<tr>
<td></td>
<td>assets</td>
</tr>
<tr>
<td></td>
<td>13.cambricon</td>
</tr>
<tr>
<td></td>
<td>15.cambricon</td>
</tr>
<tr>
<td></td>
<td>23.cambricon</td>
</tr>
<tr>
<td></td>
<td>24.cambricon</td>
</tr>
<tr>
<td></td>
<td>2.cambricon</td>
</tr>
</tbody>
</table>

• Download at runtime

/sdcard/toffee/facemodels

/$APK_HOME/cache/temp_pies/illegal_beauty2.pie
Why Local AI Models?

- AI processing is a new trend in the mobile world
  - Qualcomm NPE, Huawei HiAI, Samsung Exynos AI

- Major driven factors
  - Delay, bandwidth, privacy
Dev Process of AI Mobile App

1. **Labelled Training Data**
2. **Design + Train**
   - Using AI Framework CAFFE, Tensor Flow
3. **Model & Parameters (weights+biases)**
4. **Model hosted in Cloud, APK use Cloud APIs** OR **Model integrated in APKs**
5. **APK**
   - AI Service
6. **Mobile Devices**
   - Run on User’s Mobile Devices

Using AI Framework CAFFE, Tensor Flow, mobile devices, AI service, and APK.
Dev Process of AI Mobile App

1. Design + Train
   - Using AI Framework CAFFE, Tensor Flow
   - Running on Developer Env

2. Model & Parameters (weights+biases)
   - Vendor SDK (Huawei HiAI, Qualcomm SNPE)

3. Model Conversion
   - Platform Optimized Model Files (HiAI, DLC)

4. Vendor SDK (Huawei HiAI, Qualcomm SNPE)
   - AI lib
   - APK
     - NPU
     - AI Service
     - Mobile Devices

Run on User’s Mobile Devices
Threat related to AI Models

- model inference attack
- model inverse attack
- membership inference attack
- reverse engineering attack
- software vulnerability
Reverse Engineering Example #1（APP-A）

Model encrypted with AES,
hardcoded key in APK
Reverse Engineering Example #1 (APP-A)

Model encrypted with AES, hardcoded key in APK

```
#!/usr/bin/env python
from Crypto.Cipher import AES
import sys

key = "49806903975947393059643938538058".decode('hex')
crypto = AES.new(key,AES.MODE_ECB,key)
data = 
with open(sys.argv[1],'rb') as f:
data = data[16:]
length = len(data)
cycle = (length + 15) / 16
for i in range(cycle):
cipher = data[i*16 : (i+1)*16].ljust(16,'\x00')
text = crypto.decrypt(cipher)
f.write(text)
```

```
v5 = (void *)operator new[](0x10u);
v6 = "49806903975947393059643938538058"
v7 = v5;
while (1) {

ptr = malloc(_R5);
fwrite(ptr, 1u, _R5, v23);
aes_decrypt(128, (int)v53, v51, SHIDWORD(v51), (int)&ptr, (int)v53);
v26 = operator new[](8u);
*(BYTE *)v26 = *(BYTE *)ptr;
```
sample outcome

| 00000000 | 35 30 30 33 00 00 00 00 | 6e 61 6d 65 3a 20 22 54 | 5003....name: "TCDCH".layer {.
| 00000010 | 43 44 43 3e 22 0a 6c 61 | 79 65 72 20 7b 0a 20 20 | name: "data". type: "Input". top: "data". input_param {.
| 00000020 | 6e 61 6d 65 3a 20 22 64 | 61 74 61 22 0a 20 20 74 | shape {.
| 00000030 | 79 70 65 3a 20 22 49 6e | 70 75 74 22 0a 20 20 74 | dim: 1.
| 00000040 | 6f 70 3a 20 22 64 61 74 | 61 22 0a 20 20 69 6e 70 | dim: 60.
| 00000050 | 75 74 5f 70 61 72 61 6d | 20 7b 0a 20 20 20 20 73 | dim: 60.
| 00000060 | 68 61 70 65 20 7b 0a 20 | 20 20 20 20 20 20 69 6d | }.
| 00000070 | 3a 20 31 0a 20 20 20 20 | 20 20 64 69 6d 3a 20 31 | . shape: bottom: "data". top: "conv1". param {
| 00000080 | 0a 20 20 20 20 20 20 64 | 69 6d 3a 20 36 30 30 0a 20 | . lr_mult: 0.
| 00000090 | 20 20 20 20 20 64 69 6d | 3a 20 36 30 30 0a 20 20 20 | decay_mult: 0.
| 000000a0 | 0a 20 20 20 20 20 20 6d | 0a 6c 61 79 65 72 20 7b | 1. param {
| 000000b0 | 0a 20 20 20 20 20 20 6d | 20 22 22 23 23 23 23 23 | . lr_mult: 0.
| 000000c0 | 0a 20 20 20 20 20 20 20 | 20 22 43 6f 6e 76 6f 6c | decay_mult: 0.
| 000000d0 | 0a 20 20 20 20 20 20 20 | 20 22 62 6f 74 74 6f 6d | . convolutilon {.
| 000000e0 | 0a 20 20 20 20 20 20 20 | 20 22 74 6f 70 3a 20 22 | on_param {.
| 000000f0 | 63 6f 6e 76 31 22 0a 20 | 20 70 61 72 61 6d 20 7b | name: "conv1"
| 00000100 | 0a 20 20 20 20 6c 72 5f | 6d 75 6c 74 3a 20 30 8a | type: "Convolution".
| 00000110 | 20 20 20 20 64 69 6d 61 | 79 5f 6d 75 6c 74 3a 20 | bottom: "data". top: "conv1". param {
| 00000120 | 31 0a 20 20 20 20 20 20 | 70 61 72 61 6d 20 20 20 | . lr_mult: 0.
| 00000130 | 20 20 20 20 20 20 20 20 | 75 6e 74 3a 20 30 30 8a | decay_mult: 0.
| 00000140 | 20 20 20 20 20 20 20 20 | 5f 6d 75 6c 74 3a 20 30 | . convoluti
| 00000150 | 0a 20 20 20 20 20 20 20 | 6f 6e 74 6f 6c 75 74 69 | on_param {.
| 00000160 | 6f 6e 5f 70 61 72 61 6d | 20 20 20 20 20 20 6e | name: "conv1"

decrypted model file (caffemodel)
How to Locate Model Files?

- Popular model loading locations
  - `Init()`, `create_handle()`, `create()`
  - check all `fopen()` calls

- Names related to DL models
  - keywords: `conv`, `sigmoid`, `layer`, `CAFFEE`
  - functions: `Forward()`
  - other clues from framework:
    - `Net::NetParameter`, `NetDef::NetDef`
Reverse Engineering Example #2 (APP-B)

Caffe::NetParameter obj entry point...
Other Clues to Locate Model Related Functions

- Often framework functions are hidden or in virtual functions
- check *specific math functions*, and trace back based on cross-references
  - how often does your APP use $exp()$ function family?
- For strong obfuscations, dynamic trace and analysis might be needed

\[ A = \frac{1}{1+e^{-x}} \]
Locate DL Functions by Dynamic Analysis

Infer functions through its memory access patterns
Locate DL Functions by Dynamic Analysis

- Convolution Layer
  
  **Kernel_size = 5*5**
  **Stride = 1**
  input has 2 channel

- Pooling Layer
  
  **Kernel_size = 2*2**
  **Stride = 2**

  **Kernel_size = 3*3**
  **Stride = 3**

Layer parameters can be inferred through memory access pattern
Model Reuse Example

- **A popular image processing application**

- **Built-In App Models**:
  - APK assets directory contains 10 local model files
  - Version with AI hardware: no encryption
  - Version without AI hardware: encryption + steganography

- **Downloaded Models**:
  - For other specific functions, App pulls models from network before their first-time use
  - Models are then saved locally
To Use Reverse Engineered Models (Step 1)

• Load model in the new App (here we use the HIAI demo)

```java
private class loadModelTask extends AsyncTask<Void, Void, Integer> {
    @Override
    protected Integer doInBackground(Void... voids) {
        int ret = ModelManager.loadModelSync("2", mgr); //load model in file “2.cambricon”
        return ret;
    }
}
```
To Use Reverse Engineered Models (Step 2)

- Extract the dimensions of input and output layers

Reversed File Format for Cambricon Models
To Use Reverse Engineered Models (Step 3)

• Adjust the input and output scales

```c
//inputtensor = HIAI_TensorBuffer_create(1, 3, 299, 299);
inputtensor = HIAI_TensorBuffer_create(1, 3, 800, 800);

HIAI_TensorBuffer *inputtensorbuffer[] = {inputtensor};

//outputtensor = HIAI_TensorBuffer_create(1, 1001, 1, 1);
outputtensor = HIAI_TensorBuffer_create(1, 3, 800, 800);

... ...

float *outputBuffer = (float *) HIAI_TensorBuffer_getRawBuffer(outputtensor);

jfloatArray result;
result = (jfloatArray)env->NewFloatArray(3 * 800 * 800);

env->SetFloatArrayRegion(result, 0, 3 * 800 * 800, outputBuffer);
```
Model “Reuse” Demo

- Porting a model to an open source demo application (no training efforts needed)
New Trend: NPU Protection
Dev Process of AI Mobile App

- Labelled Training Data
- Design + Training
- Model & Parameters (weights + biases)
- Model Conversion
- Platform Optimized Model Files (HiAI, DLC)
- AI lib
- APK
- NPU
- Mobile Devices

- Using AI Framework CAFFE, Tensor Flow
- Running on Developer Env
- Vendor SDK (Huawei HiAI, Qualcomm SNPE)
- Run on User’s Mobile Devices
Example of SNPE Dev Process

- Model Conversion from Caffe to DLC
  
  ```
  % snpe-caffe-to-dlc --caffe_txt $SNPE_ROOT/models/alexnet/caffe/deploy.prototxt \\
  --caffe_bin $SNPE_ROOT/models/alexnet/caffe/bvlc_alexnet.caffemodel --dlc alexnet.dlc
  ```

- We developed a tool to convert DLC back to Caffe model
  
  ```
  % python dlc-to-caffe.py -i dlcfile -o textmodel
  % ./dump_t2m textmodel caffemodel_file
  ```
Dev Process of AI Mobile App

1. **Labelled Training Data**
   - Using AI Framework CAFFE, Tensor Flow
   - Running on Developer Env

2. **Design + Train**
   - Model & Parameters (weights+biases)

3. **Model Conversion**
   - Encrypt
   - Platform Optimized Model Files (HiAI, DLC)

4. **Vendor SDK (Huawei HiAI, Qualcomm SNPE)**

5. **AI Service**
   - APK
   - NPU
   - Mobile Devices
   - Run on User's Mobile Devices
The Current NPU Protection is Weak.

Its current “encryption” can be easily reversed (details omitted).

A real encryption + NPU HW decryption can be much stronger.
What about Cloud AI Services?
Threat related to AI Models

- model inference attack
- model inverse attack
- membership inference attack
- reverse engineering attack
- software vulnerability
My Prior Work

AI App Security Risk

- Implementation Security
  - Model Security
    - Adversarial ML
    - Model Backdoor
    - Model Theft
    - Sensor Security
    - Flaws in Framework
  - Logical Flaws
  - Data Integrity Security
    - Data Poisoning
    - Scaling Attack
    - Risk over Network
CatInPic Deep Learning Demo App

Cloud service connects back, provides a remote shell

upload a crafted image (carried shellcode)
CVE 2017-12603 : Heap Overflow

```cpp
bool BmpDecoder::readHeader()
{
    if( size >= 36 )
    {
        m_width = m_strm.getDWord();
        m_height = m_strm.getDWord();
        m_bpp = m_strm.getDWord() >> 16;
        m_rie_code = (BmpCompression)m_strm.getDWord();
        m_strm.skip(12);
        int clrused = m_strm.getDWord();
        m_strm.skip( size - 36 );

        if( m_width > 0 && m_height != 0 && ......
            (m_bpp == 8 && m_rie_code == BMP_RLE8))
        {
            iscolor = true;
            result = true;

            if( m_bpp <= 8 )
            {
                memset(m_palette, 0, sizeof(m_palette));
                m_strm.getBytes(m_palette, (clrused == 0 ? 1<<m_bpp : clrused)*4);
                iscolor = IsColorPalette( m_palette, m_bpp );
            }
            else if ...
        }
    }
}
```

```cpp
int RByteStream::getBytes( void* buffer, int count )
{
    uchar* data = (uchar*)buffer;
    int readed = 0;
    assert( count >= 0 );

    while( count > 0 )
    {
        int l;

        for(;;)
        {
            l = (int)(m_end - m_current);
            if( l > count ) l = count;
            if( l > 0 ) break;
            readBlock();
            memcpy( data, m_current, l );
            m_current += l;
            data += l;
            count -= l;
            readed += l;
        }
        return readed;
    }
}
```
Software at the Cloud side

- Caffe
- CPPClassification
- Model: BAIR/BVLC CaffeNet Model

Downloaded from GitHub on Oct 25, 2017
A query with image data is sent to the cloud side. The cloud side runs prediction using the CaffeNet model and returns a prediction result of 0.6 - "cat". The software used includes Caffe, CPPClassification, and the Model: BAIR/BVLC CaffeNet Model. This diagram was downloaded from GitHub on Oct 25, 2017.
Software at the Cloud side

- Caffe
- CPPClassification
- Model: BAIR/BVLC CaffeNet Model

Downloaded from GitHub on: Oct 25, 2017
Software at the Cloud side

- Caffe
- CPPClassification
- Model: BAIR/BVLC CaffeNet Model

Downloaded from GitHub on: Oct 25, 2017
# Steps to Build Caffe Deep Learning Image Classifier
# Instruction: http://caffe.berkeleyvision.org/Installation.html
# Dependencies: OpenCV, OpenBlas

- OpenCV (latest stable version as of Sep 28, 2017)
  https://github.com/opencv/opencv/archive/2.4.13.4.zip
  cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo ..
  sudo make install

- Openblas
  git clone https://github.com/xianyi/OpenBLAS.git
  make; sudo install

- Caffe
  git clone https://github.com/BVLC/caffe.git
  sudo apt-get install libprotobuf-dev libsnappy-dev libopencv-dev
  libhdf5-serial-dev protobuf-compiler
  sudo apt-get install --no-install-recommends libboost-all-dev
  make all

- Web Service based on Django
  pip install -U Django (1.10, 1.11 has been tested)
  python manage.py runserver 0.0.0.0:8000

---

# Model Info
Name: BAIR/BVLC CaffeNet Model
Caffemodel:
  bvlc_reference_caffenet.caffemodel
Caffemodel_url: http://dl.caffe.berkeleyvision.org/
  bvlc_reference_caffenet.caffemodel
Caffe_commit:
  709dc15af4a06bebda027c1eb2b3f3e3375d5077
AI Application Threats Landscape (Kang’s view)

AI App Security Risk

- Model Security
  - Adversarial ML
  - Model Backdoor
  - Model Theft

- Implementation Security
  - Sensor Security
  - Flaws in Framework
  - Logical Flaws

- Data Integrity Security
  - Data Poisoning
  - Scaling Attack
  - Risk over Network
Summary

- Models and parameters are valuable AI assets
- Endpoint HW capability makes local models increasingly popular
- Local models are vulnerable to reverse engineering
- DL frameworks present strong clues to defeat simple obfuscations
- Software vulnerabilities pose threats to cloud models
Q&A
kangli.ctf@gmail.com