DNN 干扰推理实验报告

实验环境

- gcc : gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
- make: GNU Make 4.2.1
- OpenBLAS: 0.3.20
- cmake: 3.16.3
- clang: ubuntu clang version 12.0.0-3ubuntu1~20.04.5

实验步骤

1. 训练 DNN 模型
   Cd poc/train
   Python prepare_dataset.py –dataset gtsrb
   Python train.py--networkvgg16—dataset gtsrb
   Python export_model.py--networkvgg16—dataset gtsrb
2. 攻击实验步骤
   1.编译 openblas

   cd poc
   make

   2.编译 llvm-pass

   下载 llvm

   cd ~
   wget https://apt.llvm.org/llvm.sh
   chmod +x llvm.sh
   sudo ./llvm.sh 12

   修改软链接

   clang --version
   which clang
   ls -l /usr/bin | grep clang
   rm /usr/bin/clang #删除已经存在的软链接
   ln -s /usr/lib/llvm-12/bin/clang /usr/bin/clang
   按照相同的方法删除并修改, clang++, opt

   编译 llvm-pass
   更改文件中的路径
   在编译之前要把 flipLauncher\flipLauncher.cpp 中 20 行和 79 行

的两个部分改为绝对路径

cd attack\source\ground_truth\llvm-pass
mkdir build && cd build
cmake ..
make

检查编译好的 llvm

修改如下文件
ground_truth/openblas_makefiles/attack/dirvier/level3/Makefile
ground_truth/openblas_makefiles/attack/interface/Makefile
用 vim 中的字符串匹配查找${HOME}/xxx/将其修改为绝对路径

替换 attack\source\ground_truth\Makefile 的 34,35,36 行中的对象
EXP_CONF = gtsrb_vgg16
EXP_DATASET = gtsrb
EXP_NETWORK = vgg16
进行攻击
cd attack\source\ground_truth
make ground_truth

具体实验内容

1. 训练模型
   准备训练集

```
ubuntu@rowhammer ~/r/S/p/train (main)> python prepare_dataset.py --dataset gtsrb
```

   若要更换训练集 修改后面参数即可

   训练模型

```
ubuntu@rowhammer ~/r/S/p/train (main)> python train.py --network vgg16 --dataset gtsrb
```

   这里选择要用的模型和训练集

   导出数据

```
ubuntu@rowhammer ~/r/S/p/train (main)> python export_model.py --network vgg16 --dataset gtsrb
```

   在具体实验过程中，直接运行代码会用尽 cpu，这里的解决办法是将作者代码中的 str 改为 numpy 自带的转化函数

```python
def export_network(args):
    model_folder = args.dataset + "_" + args.network
    model_abs_path = os.path.join(args.save_root, model_folder)
    model_path = os.path.join(model_abs_path, args.best_name)
    model = torch.load(model_path, map_location=torch.device('cpu'))
    model = model.state_dict()
    np.set_printoptions(precision=8, threshold=sys.maxsize)
    output_path = os.path.join(model_abs_path, args.output_name)
    with open(output_path, "w") as out_f:
        for name in model:
            value = model[name].cpu().numpy()
            out_f.write(f"{name} {value.shape}\n")
            out_f.write(str(value) + "\n")
```

修改为

```python
with open(output_path, "a") as out_f:
    out_f.write(f"{name} {value.shape}\n")
    np.set_printoptions(precision=8, threshold=sys.maxsize)
    out_f.write(np.array2string(value, separator=' ', formatter={'all': lambda x: f'{x:.8e}'}))
    out_f.write("\n")
```

2.配置 ml 接口

Cd poc

Make //不需要 make install

```
ubuntu@rowhammer ~/r/S/poc (main)> make
```

安装 llvm

```
a         Documents/  frp/      miniconda3/                                Music/     Public/     snap/       Videos/
Desktop/  Downloads/  llvm.sh*  Miniconda3-py39_24.11.1-0-Linux-x86_64.sh  Pictures/  rowhammer/  Templates/  zen.tar
ubuntu@rowhammer ~> wget https://apt.llvm.org/llvm.sh                                                        (base)
```

```
ubuntu@rowhammer ~> sudo ./llvm.sh 12
```

修改软链接

查找

```
ls -l /usr/bin | grep clang++
```

删除旧连接

```
rm /usr/bin/clang++
```

更新新链接

```
ln -s /usr/lib/llvm-12/bin/clang++ /usr/bin/clang++
```

一定要把三个路径全修改了

3.编译 llvm-pass

1. 修改为绝对路径

```
ubuntu@rowhammer ~> cd rowhammer/SGXBLAS/attack/source/ground_truth/llvm-pass/flipLauncher/
ubuntu@rowhammer ~/r/S/a/s/g/l/flipLauncher (main)> ls
CMakeLists.txt*  flipLauncher.cpp*
ubuntu@rowhammer ~/r/S/a/s/g/l/flipLauncher (main)> vim flipLauncher.cpp
ubuntu@rowhammer ~/r/S/a/s/g/l/flipLauncher (main)>
```

```
        raw_fd_ostream info_file_out(BRANCH_INFO_FILE, EC, sys::fs::OF_Append);
        std::string command_prefix = "opt -enable-new-pm=0 -load /home/ubuntu/rowhammer/SGXBLAS/attack/source/ground_tru
th/llvm-pass/build/flipBranches/libflipBranches.so -flipbranches -o ";
```

```
#ifndef BRANCH_INFO_FILE
#define BRANCH_INFO_FILE "/home/ubuntu/rowhammer/SGXBLAS/attack/source/ground_truth/br_info.tmp"
```

2. 编译

cd attack\source\ground_truth\llvm-pass

mkdir build && cd build

cmake ..

make

3.进行攻击

修改 makefile

ground_truth/openblas_makefiles/attack/dirvier/level3/Makefile

ground_truth/openblas_makefiles/attack/interface/Makefile

```
ubuntu@rowhammer ~> cd rowhammer/SGXBLAS/attack/source/ground_truth/openblas_makefiles/attack/driver/level3/     (base)
ubuntu@rowhammer ~/r/S/a/s/g/o/a/d/level3 (main)> ls                                                          (base)
Makefile*
```

在 makefile 中需要修改的不止一处，建议用“/SGXBLAS”来匹配所有的待修改处

替换 attack\source\ground_truth\Makefile 的 34,35,36 行中的对象

EXP_CONF = gtsrb_vgg16

EXP_DATASET = gtsrb

EXP_NETWORK = vgg16

Cd ground_truth

Make init

Make ground_truth

实验结果:

```
ubuntu@rowhammer ~/r/S/a/s/ground_truth (main)> grep -r "*3.8" results/
results/gtsrb_vgg16/br8/log.txt:*3.8005%
ubuntu@rowhammer ~/r/S/a/s/ground_truth (main)> grep -r "*5" results/
results/gtsrb_vgg16/br9/log.txt:*5.7007%
results/gtsrb_vgg16/br10/log.txt:*5.7007%
results/gtsrb_vgg16/br15/log.txt:*5.7007%
results/gtsrb_vgg16/br14/log.txt:*5.7007%
results/gtsrb_vgg16/br16/log.txt:*5.7007%
results/gtsrb_vgg16/br18/log.txt:*5.7007%
results/gtsrb_vgg16/br11/log.txt:*5.7007%
results/gtsrb_vgg16/br5/log.txt:*5.7007%
results/gtsrb_vgg16/br13/log.txt:*5.7007%
results/gtsrb_vgg16/br17/log.txt:*5.7007%
results/gtsrb_vgg16/br12/log.txt:*5.7007%
results/gtsrb_vgg16/br19/log.txt:*5.7007%
```

分支 5,9-19 在干扰下推理正确率为 5.7%，分支 8 在攻击下模型准确率 3.8%，正常推理准确率为 91%

实验过程中的问题:

1. 模型导出过程中，由于项目原先给出的代码在 32g 及一下的 cpu 中无法运行，故需要更改代码，将其 str 改为 numpy 自带的转化函数

```
def export_network(args):
    model_folder = args.dataset + "_" + args.network
    model_abs_path = os.path.join(args.save_root, model_folder)
    model_path = os.path.join(model_abs_path, args.best_name)
    model = torch.load(model_path, map_location=torch.device('cpu'))
    model = model.state_dict()
    np.set_printoptions(precision=8, threshold=sys.maxsize)
    output_path = os.path.join(model_abs_path, args.output_name)
    with open(output_path, "w") as out_f:
        for name in model:
            value = model[name].cpu().numpy()
            out_f.write(f"{name} {value.shape}\n")
            out_f.write(str(value) + "\n")
```

修改为

```
with open(output_path, "a") as out_f:
    out_f.write(f"{name} {value.shape}\n")
    np.set_printoptions(precision=8, threshold=sys.maxsize)
    out_f.write(np.array2string(value, separator=' ', formatter={'all': lambda x: f'{x:.8e}'}))
    out_f.write("\n")
```

2. 安装 ML 接口，作者 md 里给出的指令是 make install,在复现过程中发现应该是 make.
3. 修改软链接过程中应该注意将 clang,clang++,opt 的路径都要修改
4. 在检查编译好的 llvm 过程中需要注意待修改的路径不止一处。
5. 进行攻击前要 make init