CAAD CTF 2018 Toolkit Readme

The CAAD CTF 2018 Toolkit includes 2 parts.

Part one is a sample defense classifier. It is in the defense folder. It will monitor an input folder. If a new image file is created in the input folder, the defense classifier will read it and save the classification result into the output folder.

To run the defense, you can run it in Docker:

1.

```
docker run --rm -ti -v /.../defense:/code -v /.../input:/input  -v
/.../output:/output  -w /code  tensorflow/tensorflow:1.1.0 /bin/bash
```

Note: you need to modify the folder path to actual ones. If you need to use GPU, the command is below:

```
nvidia-docker run --rm -ti -v /.../defense:/code -v /.../input:/input  -v
/.../output:/output  -w /code  tensorflow/tensorflow:1.1.0-gpu
/bin/bash
```

2. After the command shell is open in Docker, go to the code folder, run the command below:

```
./run_defense2.sh /input /output
```

Note: if you don’ t run the defense in Docker, then directly use this command, bypassing step 1.

Part two is a wrapper. The wrapper can listen on port 8888, waiting requests from attackers then respond with results.

To start it, run command:
python defensewrapper.py

The wrapper supports commands below: (You may need to change the ip address when you do test)

Before the competition, each team will be assigned an attackid. The attackid will be used to identify different teams. It should be kept secret. Otherwise others can act as your team. The attackid should be attached in each command sent to wrapper.

1. getmydefense. Send this command to wrapper, you will get your team’ s defense id. You cannot attack your own defense.

GET
127.0.0.1:8888/?cmd=getmydefense&attackid=12345
Response sample: (json format)

```
{
    "msg": "getmydefense succeed",
    "classlabel": 123,
    "defenseid": 1,
    "result": 0
}
```


2. getdefenses. This command will respond with all the defenses you can attack.

Get
127.0.0.1:8888/?cmd=getdefenses&attackid=12345
Response sample: (json format)


```
[
    {
        "classlabel": 260,
        "defenseid": 2
    },
    {
        "classlabel": 851,
        "defenseid": 3
    },
    {
        "classlabel": 74,
        "defenseid": 4
    },
    {
        "classlabel": 309,
        "defenseid": 5
    }
]
```

3. getteamlogo. This command will respond the image which the team need to use as the source image. Create adversarial examples based on this image to attack other teams.

GET
127.0.0.1:8888/?cmd=getteamlogo&attackid=12345
Response: binary data content of .png file.

4. postattack. This command will be used to attack other defenses. The defenseid parameter in the request is the target defense. The epsilon parameter is the perturbation. The epsilon value can be 1,2,4,8,16,32

POST
172.16.2.116:8888/?cmd=postattack&attackid=abcde&defenseid=2&epsilon=32
Post data body: binary data content of adversarial example .png file.

Response sample: json format

```
{
   "msg": "attack performed",
   "classlabel": 34,
   "attackresult": "negative",
   "result": 0,
   "description": "loggerhead, loggerhead turtle, Caretta caretta\n"
}
```

If attack is successful, the “attackresult” field will be positive. Otherwise, it is negative. The classlabel is the class the defense put the image in. Result is 0 if attack performed, otherwise there are problems when attacking.